LITHIUM: Making Iterative Rejection Sampling Practical for Compact Lattice Signatures

Jipeng Zhang[1], Pengfei Chen[2], Long Chen[3], Cong Zhang[2], Jiaheng Zhang[1]

[1]National University of Singapore [2]Zhejiang University [3]Institute of Software, Chinese Academy of Sciences

jp-zhang@outlook.com, chenpengfei155@gmail.com, chenlong@iscas.ac.cn, 0022090@zju.edu.cn, jhzhang@nus.edu.sg

Abstract

Post-quantum deployments need signatures that are both fast and small. ML-DSA gives a practical Fiat-Shamir lattice-signature baseline, but its signatures remain large enough to make bandwidth, certificate size, and signed-log storage first-order costs. Gärtner's iterative rejection sampling construction (CRYPTO'25) shows that this design family can be made much more compact. The open question is whether this theoretical design can be turned into a concrete, implementation-oriented signature scheme, where the parameters, algorithms, encodings, and optimized software work together without giving up the promised compactness.

We present LITHIUM, a compact Fiat-Shamir lattice signature that makes iterative rejection sampling practical. LITHIUM co-designs its parameters, discrete Gaussian sampler, ApproxExp evaluation, iterative rejection sampling, and rANS encoder so that compact signatures do not come at the cost of an impractical signer. For the core components, we introduce algorithmic and vectorized optimizations and provide a portable reference implementation together with vectorized AVX2 and AVX-512 implementations. LITHIUM-120 targets a security level close to ML-DSA-44. Our experiments show that our fastest implementation signs faster than ML-DSA-44 while producing signatures about half as large: 1,187 bytes versus 2,420 bytes. Compared with HAETAE-120, LITHIUM-120 is more compact and signs about 6.8× faster.

1 Introduction

Lyubashevsky introduced Fiat-Shamir with aborts as a way to turn lattice identification protocols into non-interactive signatures while preserving a clean distributional argument for secrecy [33, 34]. In this paradigm, a signer samples a fresh mask, derives a Fiat-Shamir challenge, and forms a response that algebraically mixes the mask with the long-term secret. Since a short lattice response cannot perfectly hide that secret contribution by itself, signing is best viewed as a retry loop: each iteration builds one candidate response, and an abort means that this candidate is rejected and the signer retries with fresh randomness. The loop stops only when rejection sampling accepts a candidate; conditioned on this acceptance event, the emitted response follows a distribution that is independent of the secret key. This mechanism has become one of the central design templates for SIS/LWE-based signatures [2, 43].

Ducas, Kiltz, Lepoint, et al. designed CRYSTALS-Dilithium [20] based on Fiat-Shamir with aborts. Its standardized form, ML-DSA, was published by NIST in FIPS 204 [36] and bases its security on Module-LWE and Module-SIS [32, 36]. ML-DSA is attractive because signing is fast, the arithmetic is simple, and constant-time software is comparatively straightforward; its cost profile, however, is not negligible, since verification keys and signatures are larger than in compact lattice alternatives such as Falcon [25].

Gärtner's CRYPTO 2025 work introduced iterative rejection sampling (IRS) as a new way to reduce this compactness gap inside the Fiat-Shamir-with-aborts family [27]. Instead of constructing one secret-shifted candidate response and deciding whether to keep it, IRS builds the response through a sequence of small rejection-sampling decisions, each associated with one challenge-selected secret component; because each step handles a shorter vector than the full secret contribution, the scheme can target a much narrower output distribution at comparable overall acceptance probability. Gärtner's compact variant, which still has a bounded rejection probability, brings the combined verification-key and signature size close to Falcon; the non-aborting variant's combined size is below 60% of ML-DSA at the corresponding security level.

This theoretical promise leaves a systems question unresolved. Gärtner's construction uses discrete Gaussian sampling, a component well known to dominate the signing cost of schemes such as Falcon and HAETAE [12]; separately, the IRS component introduces infinite-series expressions that require many exponential evaluations. These two ingredients are exactly the kind of arithmetic that can turn an elegant compact-signature theorem into a slow or hard-to-audit implementation. This paper therefore asks:

> *Can iterative rejection sampling be used to build a lattice signature that is simultaneously fast, securely implementable in software, and compact?*

Table 1: Overview at the ML-DSA-44 security level.

Scheme	vk	sig	vk+sig	time
LITHIUM-**120**	1056 B	1187 B	**2243 B**	**166 k**
ML-DSA-44 [36]	1312 B	2420 B	3732 B	191 k
HAETAE-120 [12]	992 B	1474 B	2466 B	1249 k
Falcon-512 [25]	897 B	666 B	1563 B	824 k
Raccoon-128 [15]	2256 B	11524 B	13780 B	5501 k

* This level corresponds to approximately 120-bit classical Core-SVP security. Here, vk is the verification key size, sig is the signature size, vk+sig is their sum, and time is the signing time measured by StQ_2 [7], the stabilized median; k denotes 1000 CPU cycles. LITHIUM-120: AVX-512IFMA version. Raccoon-128: ref version. Others: AVX2 versions.

Table 1 previews our answer. For size and speed comparisons, we use UF-CMA/search-security estimates as the primary matching axis and report strong-unforgeability estimates separately in Table 2. Under this matching, LITHIUM-120 keeps the verification key and signature sum below HAETAE-120 and far below ML-DSA-44; its signature is roughly half the size of ML-DSA-44. This is the goal reflected in the name: relative to Dilithium, LITHIUM is a lighter Fiat-Shamir lattice signature. Our fastest implementation is faster than the ML-DSA-44 AVX2 baseline on our benchmark platform. The table is not meant to collapse all engineering tradeoffs into one number: Falcon follows the GPV hash-and-sign approach [25, 28], Raccoon targets masking-friendly signing [15, 16], and the schemes differ in standardization status and side-channel surface. It does show the central point of this work: compact Fiat-Shamir signatures need not be impractical.

Beyond compactness, LITHIUM also avoids two issues seen in the most prominent lattice signatures.

The long signing-time tail of ML-DSA. Median and mean are the usual summaries of signing latency, but ML-DSA exhibits a long signing-time tail that practitioners have found does not capture the full picture [39, 45]. FIPS 204 reports 4.25 expected repetitions, a 76.5% per-attempt rejection probability, for ML-DSA-44 signing [36], and in our benchmark p99 signing time reaches roughly four times p50.[1] LITHIUM avoids this tail by targeting a signing retry probability near 2^{-100}.

The vectorization incompatibility of Falcon. Accelerating cryptography with AVX2 is mature practice, e.g., NewHope at USENIX Security 2016 [5], and AVX-512 accelerates it further, e.g., ENG25519 at USENIX Security 2024 [49]; LITHIUM's byte format is independent of vector layout, so future wide-vector backends can reuse the same dispatch boundary after ordinary validation. Falcon's costly discrete Gaussian sampler benefits substantially from such vectorization [50], but its algorithm stabilized around 2020 and cannot keep its KATs compatible with these vectorized implementations. We designed LITHIUM with this lesson in mind: its reference, AVX2, and AVX-512 backends share one byte-level specification and pass the same KATs.

[1] Under the benchmark conditions of this work, the p99 signing time is 758k CPU cycles, while the p50 is 186k cycles.

1.1 Technical Overview

LITHIUM pursues three targets: fast, securely implementable, and compact signing.

Fast signing. LITHIUM picks parameters so the signing retry probability is near the 2^{-100} target, avoiding long-tail latency, and shapes the algorithmic flow around CPU vector instructions, as is now mature practice for ML-KEM (FIPS 203 [37]) and ML-DSA (FIPS 204 [36]). Its reference, AVX2, and AVX-512 backends share one byte-level specification and pass the same KATs, avoiding the vectorization-compatibility issues seen in Falcon, and the dominant components—Gaussian sampling, ApproxExp, and IRS decisions—are optimized both algorithmically and through vectorization.

Secure implementation. By *secure implementation* we mean the standard constant-time discipline—the secret-dependent signing core avoids secret-dependent branches, memory accesses, division, square roots, and gathers—targeting timing and cache leakage, not physical side channels or faults. We combine source-level constant-time programming with assembly-level scans and with dudect-style tests, following lessons from KyberSlash [9] and recent ML-DSA bugs [8].

Compactness. LITHIUM's compactness comes from Gärtner's iterative rejection sampling [27]. Turning that route into a concrete scheme needs more than parameters: the IRS series must be truncated with a concrete error bound, the many exponentials must be fast, Gaussian sampling must drop from a bottleneck to a practical cost, and the response must be encoded compactly without meaningful failure probability—a design strategy and component optimizations that may help future lattice signatures.

1.2 Contributions

This paper makes the following contributions.

- We instantiate Gärtner's IRS approach as LITHIUM, a concrete lattice signature with implementation-oriented parameters, pre-NTT public data, fixed byte encodings, and a signing retry budget at the 2^{-100} scale.
- We account for the approximation steps that an implementation must introduce: the discrete Gaussian sampler, ApproxExp evaluation, the forced two-output IRS decision, finite IRS-series truncation, and fixed-size rANS encoding.

- We build optimized signing components for the practical bottlenecks of IRS-based signatures, with portable reference code and AVX2/AVX-512 backends whose default paths preserve KAT-compatible integer behavior and sign-verify interoperability.
- We evaluate LITHIUM against mainstream lattice signatures. LITHIUM-120 has a 1056-byte verification key and a 1187-byte signature; its default AVX2 signer has near-one tail, and its fastest opt-in signer is slightly faster than the AVX2-optimized ML-DSA-44.

Our implementations are publicly available at https://github.com/Ji-Peng/Lithium-Code-Artifact.

2 Background

This section fixes the notation and implementation scope used by the rest of the paper. We keep the background narrow: polynomial modules, the Fiat-Shamir-with-aborts template, Gärtner's iterative rejection sampling (IRS) [27], x64 vector backends, and the distributional/error notions that become implementation obligations in LITHIUM.

2.1 Notation and Polynomial Modules

For a power-of-two degree n and a prime modulus q, let $R = \mathbb{Z}[X]/(X^n+1)$ and $R_q = R/qR$. A polynomial is written in lowercase, a vector of polynomials in bold lowercase, and a matrix of polynomials in bold uppercase. Unless stated otherwise, vectors and matrices are over R or R_q with dimensions clear from context. For $\mathbf{a} \in R^d$, $\|\mathbf{a}\|$ is the Euclidean norm over all dn integer coefficients and $\|\mathbf{a}\|_\infty$ is the coefficient-wise infinity norm. We write $\mathcal{D}_{\mathbb{Z},\sigma}$ for the centered discrete Gaussian over integers and extend it coefficient-wise to polynomials and polynomial vectors.

Polynomial products use an NTT representation. Let $\mathsf{NTT} : R_q \to T_q$ be the number-theoretic transform and let $\circ$ denote coefficient-wise multiplication in T_q. As in Dilithium [20], uniformly sampled public matrix material may be generated directly in the NTT domain because NTT is a bijection on R_q. LITHIUM uses this pre-NTT idea for the public matrix $\widehat{\mathbf{A}}_0 \in T_q^{m\times\ell}$, but it does not turn the whole scheme into an NTT-domain protocol. Responses, hints, norm checks, low bits, and challenge shifts remain in the coefficient domain because their distributions and encodings are coefficient-wise.

The secret key has the block form $\mathbf{s} = [f, \mathbf{s}_0, \mathbf{e}]^\mathsf{T} \in R^{1+\ell+m}$, where $f = 2f_0 + 1$ and $f_0, \mathbf{s}_0, \mathbf{e}$ are narrow Gaussian samples. The public key is an NTWE-style vector following Gärtner's construction [27]:

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

NTWE combines an LWE-like noisy inner product $\bar{\mathbf{a}} \cdot \bar{\mathbf{s}} + e$ with an NTRU-like multiplication by the inverse of a short invertible ring element f [26]. Its decision assumption says that $(\mathbf{A}_0, \mathbf{b})$, which gives multiple NTWE samples sharing the same short $(\mathbf{s}_0, f)$ and using independent error entries from $\mathbf{e}$, is computationally indistinguishable from uniformly random. The IRS security argument uses this pseudorandomness together with the SIS-style forgery bound, while the public key remains compact like an NTRU-flavored construction and retains the module-parameter flexibility of LWE-style designs. The verifier uses the implicit matrix

$$\begin{aligned} \mathbf{A} &= [\,q\mathbf{j} - 2\mathbf{b} \mid 2\mathbf{A}_0 \mid 2\mathbf{I}_m\,] \in R_q^{m\times(1+\ell+m)}, \\ \mathbf{A}\mathbf{s} &\equiv q\mathbf{j} \pmod{2q}. \end{aligned} \tag{2}$$

where $\mathbf{j} = [1, 0, \ldots, 0]^\mathsf{T} \in R^m$ is the unit vector and $\mathbf{I}_m$ is the $m \times m$ identity matrix over R. Thus, if a response changes from $\mathbf{y}$ to $\mathbf{y} \pm \mathbf{s}c$, the commitment changes from $\mathbf{A}\mathbf{y}$ to $\mathbf{A}\mathbf{y} \pm q\mathbf{j}c$ modulo $2q$; since $q\mathbf{j}c$ has order two modulo $2q$, the two signs give the same public correction term.

2.2 x64 Vector Backends

LITHIUM targets a scalar reference and two vectorized backends. AVX2 provides 256-bit integer vectors and is the mature baseline for high-speed cryptographic software, while AVX-512 widens the same style of arithmetic to 512-bit vectors and mask registers. AVX-512IFMA is an AVX-512 subset that supports eight-lane 52-bit integer fused multiply-add instructions, which are useful for ApproxExp.

2.3 Fiat-Shamir with Aborts and IRS

Lyubashevsky's Fiat-Shamir-with-aborts framework samples a fresh mask $\mathbf{y}$, computes a commitment $\mathbf{w}$, hashes $\mathbf{w}$ and the message into a sparse challenge c, and forms a response of the form $\mathbf{y} + \mathbf{s}c$ [33, 34]. We write κ for the number of nonzero coefficients in c. Without rejection sampling, this is a unimodal Gaussian shifted by the secret contribution $\mathbf{s}c$. Rejection sampling turns signing into a loop: candidate responses that would leak too much about the shift are rejected, and the signer retries until the accepted response follows a target distribution independent of the secret. The compactness problem is that a narrower target distribution gives shorter signatures but raises the retry probability.

BLISS changed this distributional shape through bimodal Gaussian rejection sampling [19]: since challenge coefficients are zero or one, any c' with $c' \equiv c \pmod 2$ still yields a valid signature under the key relation in Equation (2), so the signer can use either $\mathbf{y} + \mathbf{s}c$ or $\mathbf{y} - \mathbf{s}c$, producing a bimodal Gaussian before rejection rather than Lyubashevsky's unimodal shifted Gaussian. This bimodal distribution is closer to the centered target, so BLISS obtains more compact signatures at comparable rejection cost.

Gärtner's IRS construction takes the same idea one step further through two core ideas [27] as described in Algorithm 1.

Algorithm 1: Iterative rejection sampling (IRS) [27]

Input: mask $\mathbf{y}$, sparse challenge c, secret vector $\mathbf{s}$
Output: response $\mathbf{z}$ or $\perp$

$\mathbf{z} \leftarrow \mathbf{y}$
Parse c as monomials $x^{j_1}, \ldots, x^{j_\kappa}$
for $i \leftarrow 1$ **to** κ **do**
  $\mathbf{v}_i \leftarrow x^{j_i}\mathbf{s}$
  $u \leftarrow \mathcal{U}([0,1))$
  $(a,b) \leftarrow (f_{\mathbf{v}_i}(\mathbf{z}), g_{\mathbf{v}_i}(\mathbf{z}))$ // Eqs (4), (5)
  if $u < a$ **then** $\mathbf{z} \leftarrow \mathbf{z} - \mathbf{v}_i$
  else if $u < a + b$ **then** $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{v}_i$
  else return $\perp$
return $\mathbf{z}$

The first is *iteration*: instead of making one global bimodal choice for $\mathbf{s}c$, IRS parses the sparse challenge as monomials $x^{j_1}, \ldots, x^{j_\kappa}$ and processes the corresponding shifted secret vectors $\mathbf{v}_i = x^{j_i}\mathbf{s}$ one at a time.

The second idea is *adaptivity*. Given the current accumulator $\mathbf{z}$ and the next vector $\mathbf{v}$, IRS does not first choose a sign and then run a fixed accept/reject test. It dynamically chooses among three outcomes: move to $\mathbf{z}-\mathbf{v}$, move to $\mathbf{z}+\mathbf{v}$, or reject and restart the outer signing loop. This adaptive choice keeps the accumulator on the target Gaussian distribution after each successful step while controlling the total retry probability.

For a Gaussian of standard deviation r, write $\rho_r(\mathbf{x}) = \exp(-\|\mathbf{x}\|^2/(2r^2))$. For the current accumulator $\mathbf{z}$, the adaptive IRS decision probabilities are defined through the alternating series

$$S_{\mathbf{v}}(\mathbf{z}) = \sum_{k\geq 0} (-1)^k \frac{\rho_r(\mathbf{z}+k\mathbf{v})}{\rho_r(\mathbf{z})}. \tag{3}$$

For the expected repetition rate $M \geq 1$, the functions used by one IRS step are

$$f_{\mathbf{v}}(\mathbf{z}) = \begin{cases} S_{\mathbf{v}}(\mathbf{z})/M, & \text{if } \langle \mathbf{z}, \mathbf{v}\rangle \geq \|\mathbf{v}\|^2, \\ (1 - S_{\mathbf{v}}(-\mathbf{z}))/M, & \text{if } \langle \mathbf{z}, \mathbf{v}\rangle < \|\mathbf{v}\|^2, \end{cases} \tag{4}$$

$$g_{\mathbf{v}}(\mathbf{z}) = \begin{cases} (1 - S_{\mathbf{v}}(\mathbf{z}))/M, & \text{if } \langle \mathbf{z}, \mathbf{v}\rangle \geq -\|\mathbf{v}\|^2, \\ S_{\mathbf{v}}(-\mathbf{z})/M, & \text{if } \langle \mathbf{z}, \mathbf{v}\rangle < -\|\mathbf{v}\|^2. \end{cases} \tag{5}$$

The value $f_{\mathbf{v}}(\mathbf{z})$ is the probability of outputting $\mathbf{z}-\mathbf{v}$, $g_{\mathbf{v}}(\mathbf{z})$ is the probability of outputting $\mathbf{z}+\mathbf{v}$, and $1 - f_{\mathbf{v}}(\mathbf{z}) - g_{\mathbf{v}}(\mathbf{z})$ is the rejection probability.

2.4 Core Algorithmic Components

Discrete Gaussian sampling at large standard deviation. LITHIUM's mask $\mathbf{y}$ generation needs centered discrete Gaussian samples at a comparatively large standard deviation, for example $r = 182$ in LITHIUM-120. We use the BLISS-like convolution sampler [19]. The idea is to factor the target standard deviation as $r = k \cdot r_s$, draw a high part x from a base distribution $\mathcal{D}_{\mathbb{Z}^+, r_s}$ of small standard deviation with BaseSampler (which scans a reverse-CDT table, adding $[\![u < t]\!]$ over thresholds t for a random word u), draw a low part y uniformly from $\{0, \ldots, k-1\}$, and combine them as $z = kx + y$. A Bernoulli rejection with acceptance probability $\exp(-y(y+2kx)/(2r^2))$ corrects the combined variable to the exact target Gaussian $\mathcal{D}_{\mathbb{Z},r}$, and a sign bit and a zero-fold make it two-sided. The base distribution has small standard deviation (empirically $r_s \leq 2$), so its reverse-CDT (RCDT) scan is short; truncating the support around 10 or 11 standard deviations and scaling the thresholds by 2^{93} meets the concrete security-error budget. We write SampleZ for this sampler and BaseSampler with an explicit random word u because the implementation generates randomness in the caller and keeps the RCDT scan as a pure table operation.

Approximating the exponential (ApproxExp). The BLISS-like sampler's Bernoulli rejection evaluates $\exp(\cdot)$ on sampler-internal variables, while the IRS decision in Algorithm 1 evaluates $\exp(\cdot)$ on inputs that additionally depend on the secret key, making it the security-critical case for ruling out data-dependent timing. We use fixed-degree polynomial approximations generated with the lattice-aware rounding of GALACTICS (CCS 2019) [6] and the real-coefficient minimax routines of the Sollya toolkit [13], which together enable a constant-time implementation with a fixed instruction trace.

rANS entropy coding. As in Gärtner's construction [27] and HAETAE [12], the components in generated signatures are far from uniform; instead, they are shaped by Gaussian distributions. We therefore also use the range variant of asymmetric numeral systems (rANS), an entropy coder introduced by Duda [23] that attains near-Shannon-optimal rates, to reduce the signature size.

3 LITHIUM Construction

This section specifies LITHIUM, including key generation (Algorithm 2), signing (Algorithm 3), and verification (Algorithm 4); the three security levels are parameterized in Table 2. The descriptions of selected subroutines appear in Appendix A. At the algorithmic level, our scheme can be viewed as a refinement of Gärtner's construction [27] with additional implementation-oriented details. Our implementation instantiates H and G as SHAKE256 and SHAKE128 respectively.

3.1 Parameter Selection

Gärtner's work provides parameter sets for the same security levels with rejection probability around 2^{-110} [27, Table 2]. Because LITHIUM changes several parameter-selection choices, we rerun the search. The basic objective follows Gärtner's search: satisfy the NTWE and SIS-UF estimates at the

Table 2: Main LITHIUM parameter sets.

Parameter	LITHIUM-120	LITHIUM-180	LITHIUM-260
n, q	$256, 57601$	$256, 30977$	$512, 64513$
ℓ, m	$2, 2$	$3, 3$	$2, 2$
κ, τ	$30, 512$	$58, 512$	$58, 1024$
σ, r	$1, 182$	$1, 204$	$1, 257$
B'_k, B_k	$44.5, 46.0$	$49.9, 51.3$	$63.2, 64.8$
B_s, B_v	$8090, 10106$	$10396, 12794$	$15210, 20646$
NTWE/SIS-UF/sUF bits	$125/125/101$	$189/190/156$	$283/268/224$
pk/sig bytes	$1056/1187$	$1472/1618$	$2080/2348$
pk+sig bytes	2243	3090	4428

† Here n, q are the ring degree and modulus; ℓ, m are module dimensions; κ is the challenge weight; τ is the rounding range used by MakeHint; σ, r are the secret and mask standard deviations; B'_k, B_k bound accepted secret key norms; B_s, B_v bound signing and verification norms; pk/sig are encoded byte sizes. The current signature sizes include rANS reserve slack; for the three security levels, this slack is 122, 157, and 161 bytes, respectively.

* Security estimates are classical lattice-estimator [3, 4] outputs for the NTWE, SIS-UF, and SIS-sUF bounds according to [27], where the rank-k NTWE is approximately estimated using the rank-$(k+1)$ Module-LWE. UF denotes existential unforgeability, while sUF denotes strong unforgeability.

target level and then minimize the combined public-key and signature size. We emphasize the following differences.

- Truncated Gaussian sampling and approximate exponentials introduce small security losses (e.g., the 2-bit loss of [30, Theorem 7]), so we require at least 5 bits of margin.
- Beyond Gärtner's upper norm bound B_k, we add a lower bound B'_k that controls the IRS truncation error (Section 4); together they give a $\approx 50\%$ key-acceptance rate, costing about one LWE-security bit but only at the one-time key generation.
- Each of the three signing-restart sources is gated at the 2^{-101} scale: the $\mathbf{z}$ norm check B_s, the ideal IRS rejection (set via r/B_k, then dropped by the implementation and budgeted as distributional bias in Section 4), and rANS overflow (with a zero-padded reserve). The verification bound B_v leaves probability at most 2^{-100} that a correctly generated signing attempt falls outside the verification norm bound, replacing Gärtner's exact correctness.
- Following recent small- and sparse-secret LWE analyses [14, 22, 29, 40], we take $\sigma \geq 1$; and q is prime with $n \mid q-1$ and below 2^{16} for fast vectorized NTT arithmetic.

Table 2 is thus a SIS-UF parameter set, and it is the instantiation studied throughout this paper. Rerunning the same search against SIS-sUF instead yields the parameter set of Table 7, which we report in Appendix B.2.5 for reference.

Algorithm 2: LITHIUM.KeyGen(ξ)

Input: seed ξ
Output: public key pk and secret key sk

$(\rho, \rho', K) \in \mathbb{B}^{32,64,64} \leftarrow \mathsf{H}(\xi||\ell||m, 160)$
$\widehat{\mathbf{A}}_0 \in T_q^{m\times\ell} \leftarrow \mathsf{ExpandA}(\rho)$ `// NTT domain`
$t \leftarrow 0$
repeat
  $(f_0, \mathbf{s}_0, \mathbf{e}) \in \mathcal{D}^{1,\ell,m}_{\mathbb{Z},\sigma} \leftarrow \mathsf{SampleS}(\rho'||t)$
  $f \leftarrow 2f_0 + 1$ and $\mathbf{s} \leftarrow [f, \mathbf{s}_0, \mathbf{e}]^\mathsf{T}$
  $t \leftarrow t + 1$
until $B'^2_k \leq \|\mathbf{s}\|^2 \leq B^2_k$ *and* f *is invertible in* R_q
$\widehat{\mathbf{b}} \leftarrow (\widehat{\mathbf{A}}_0 \circ \mathsf{NTT}(\mathbf{s}_0) + \mathsf{NTT}(\mathbf{e})) \circ \mathsf{NTT}(f)^{-1}$
$pk \leftarrow \mathsf{pkEncode}(\rho, \widehat{\mathbf{b}})$
$sk \leftarrow \mathsf{skEncode}(\rho, K, \widehat{\mathbf{b}}, \mathbf{s})$
return (pk, sk)

3.2 Key Generation

Key generation follows Gärtner's NTWE relation in Equation (2). As in ML-DSA [20], LITHIUM uses a pre-NTT public matrix: the seed ρ expands directly to $\widehat{\mathbf{A}}_0 \in T_q^{m\times\ell}$, and the public vector $\mathbf{b}$ is serialized in a canonical NTT order. Each optimized backend imports that order into its private multiplication layout, while IRS decisions and encodings remain in the coefficient domain.

3.3 Signing and Verification

A signing attempt samples the full mask once. IRS then walks the challenge support and chooses the sign of each shifted secret vector. A signing attempt can abort and restart in two places: the public norm gate may reject $\mathbf{z}$, and the rANS encoder may overflow its fixed reserve. The ideal IRS step could also reject, but its per-signature rejection probability is only 2^{-101}, so the implemented step always outputs and the removed restart mass is budgeted as distributional bias in Section 4. The parameters keep each component probability around 2^{-101}, so the overall signing restart probability stays at the 2^{-100} scale.

Verification reconstructs the public commitment from $(\tilde{c}, \mathbf{h}, \mathbf{z}_1)$ and accepts only if the hash and norm checks pass. Interoperability is a construction invariant: the scalar, AVX2, and AVX-512 default builds use the same canonical public-key order, byte-level encodings, and KATs, and the optional fast ApproxExp variants still output signatures the default verifier accepts because verification depends only on the public relation and canonical encoding.

Algorithm 3: LITHIUM.Sign(sk, M, rnd)

Input: secret key sk, message M, randomness rnd
Output: signature σ

$(\rho, K, \widehat{\mathbf{b}}, \mathbf{s}) \leftarrow \mathsf{skDecode}(sk)$
$\widehat{\mathbf{A}}_0 \in T_q^{m\times\ell} \leftarrow \mathsf{ExpandA}(\rho)$ `// NTT domain`
$\widehat{\mathbf{A}} \leftarrow [\mathsf{NTT}(q\mathbf{j}) - 2\widehat{\mathbf{b}} \mid 2\widehat{\mathbf{A}}_0 \mid 2\mathsf{NTT}(\mathbf{I}_m)]$
$tr \leftarrow \mathsf{H}(\mathsf{pkEncode}(\rho, \widehat{\mathbf{b}}), 64)$
$\mu \leftarrow \mathsf{H}(tr \| M, 64)$
$\rho'' \leftarrow \mathsf{H}(K \| rnd \| \mu, 64)$
$t \leftarrow 0$
repeat
  $\mathbf{y} \in \mathcal{D}_{\mathbb{Z},r}^{1+\ell+m} \leftarrow \mathsf{SampleY}(\rho'', t)$
  $t \leftarrow t + 1$
  $\mathbf{w} \leftarrow \mathsf{NTT}^{-1}(\widehat{\mathbf{A}} \circ \mathsf{NTT}(\mathbf{y})) \bmod 2q$
  $\tilde{c} \leftarrow \mathsf{H}(\mathsf{HighBits}(\mathbf{w}) \| \mathsf{LSB}(\mathbf{w}) \| \mu)$
  $c \in R_q \leftarrow \mathsf{SampleC}(\tilde{c})$
  $\mathbf{z} \leftarrow \mathsf{IRS}(\mathbf{y}, c, \mathbf{s})$ `// κ forced steps of Algorithm 6`
  if $\|\mathbf{z}\|^2 > B_s^2$ **then continue**
  $(\mathbf{z}_1, \mathbf{z}_2) \leftarrow ((z_0, \ldots, z_\ell), (z_{\ell+1}, \ldots, z_{\ell+m}))$
  $\mathbf{h} \leftarrow \mathsf{MakeHint}(\mathbf{w}, \mathbf{z}_2)$
  $\sigma \leftarrow \mathsf{sigEncode}(\tilde{c}, \mathbf{h}, \mathbf{z}_1)$
until $\sigma \neq \perp$
return σ

Algorithm 4: LITHIUM.Verify(pk, M, σ)

Input: public key pk, message M, signature σ
Output: accept or reject

$(\rho, \widehat{\mathbf{b}}) \leftarrow \mathsf{pkDecode}(pk)$
$D \leftarrow \mathsf{sigDecode}(\sigma)$
if $D = \perp$ **then return** reject
$(\tilde{c}, \mathbf{h}, \mathbf{z}_1) \leftarrow D$
$tr \leftarrow \mathsf{H}(pk, 64)$ and $\mu \leftarrow \mathsf{H}(tr \| M, 64)$
$c \in R_q \leftarrow \mathsf{SampleC}(\tilde{c})$
$\widehat{\mathbf{A}}_0 \in T_q^{m\times\ell} \leftarrow \mathsf{ExpandA}(\rho)$ `// NTT domain`
$\widehat{\mathbf{A}}_1 \leftarrow [\mathsf{NTT}(q\mathbf{j}) - 2\widehat{\mathbf{b}} \mid 2\widehat{\mathbf{A}}_0]$
$\widetilde{\mathbf{w}} \leftarrow \mathsf{NTT}^{-1}(\widehat{\mathbf{A}}_1 \circ \mathsf{NTT}(\mathbf{z}_1)) - qc\mathbf{j} \bmod 2q$
$w_0' \leftarrow \mathsf{LSB}(\mathbf{z}_1[0] - c)$ `// z1[0]: the first polynomial of z1`
$(\mathbf{w}_h', \mathbf{z}_2') \leftarrow \mathsf{UseHint}(\widetilde{\mathbf{w}}, \mathbf{h}, w_0')$
$\tilde{c}' \leftarrow \mathsf{H}(\mathbf{w}_h' \| w_0' \| \mu)$
if $\tilde{c}' \neq \tilde{c}$ **then return** reject
if $\|(\mathbf{z}_1, \mathbf{z}_2')\|^2 \leq B_v^2$ **then return** accept
return reject

4 Security Bounds

LITHIUM follows the overall structure of Gärtner's construction, so we rely on Gärtner's analysis for the signature scheme's security [27]. Practical implementation introduces a separate question: discrete Gaussian sampling and the IRS series $S_{\mathbf{v}}$ in Equation (3) cannot realize perfect distributions or infinite sums. This section proves that, under our concrete parameter choices, the approximate implementation remains indistinguishable from the ideal one. Full derivations are in Appendix B.

4.1 Approximation Security Analysis

Secret-key sampling in key generation calls SampleS, which in turn calls BaseSampler. Its effect on the LWE security estimate is through the realized secret standard deviation; for the $\sigma = 1$ BaseSampler implementation, this standard deviation differs from σ by about 2^{-23}, so the impact on the estimated LWE security is negligible.

Two implementation components require explicit distributional analysis. Signing calls SampleY, which uses SampleZ and therefore combines BaseSampler with ApproxExp in the Bernoulli correction. The IRS then truncates the infinite series $S_{\mathbf{v}}$ in Equation (3). The SampleZ statement specializes the isochronous-sampling analysis of [30] to the distribution of the emitted sample, which is all that signing reveals; the IRS statement is new to this work and analyzes the security loss caused by truncating the IRS series under LITHIUM's key-norm window. We write $R_a(P, Q)$ for the multiplicative order-a Rényi divergence,

$$R_a(P, Q) = \left(\sum_x \frac{P(x)^a}{Q(x)^{a-1}} \right)^{1/(a-1)},$$

with the usual absolute-continuity requirement $P \ll Q$.

Gärtner states that using only the first 20 terms of the IRS sum gives a value whose error is far below 2^{-256} even in the worst case $\|\mathbf{v}\| = 1$ [27, Section 3.3]. A direct endpoint calculation shows that this is not the relevant bound for our parameter range. We therefore use Theorem 2, whose proof evaluates the truncation error under the key-norm window used by LITHIUM.

Following [30], a query to a distribution means one invocation of the corresponding ideal sampler, answered by an independent sample from that distribution; the implementation replaces these invocations by the concrete samplers analyzed below. For SampleZ, the object in this hybrid is the emitted sample: its internal Bernoulli bits and proposal-loop count depend only on fresh XOF-derived signing randomness and public parameters, not on the long-term secret, and even if that count is exposed it is independent of the UF-CMA witness and does not change the emitted-sample distribution analyzed below.

The two theorems below compose by a standard hybrid argument for the signing distribution. Replacing the ideal Gaussian sampler in SampleY by SampleZ loses at most two bits by Theorem 1: one for BaseSampler and one for

ApproxExp. The IRS statement matches what the C code executes: LITHIUM operates in the negligible-rejection regime, where the ideal IRS step would output the restart symbol $\perp$ only with pointwise probability $(M-1)/M \approx 2^{-107}$, so Algorithm 6 omits the restart branch and assigns this residual mass to the $\mathbf{y}+\mathbf{v}$ output. Replacing the exact three-output IRS step by this implemented *forced* two-output step, including the series truncation, loses at most one further bit by Theorem 2. Thus, under the budgets in Table 3, the implemented signer preserves the SIS-based signing security up to a total three-bit loss.

Relation to Gärtner's negligible-rejection variant. Gärtner already shows that parameter sets with negligible rejection can drop all rejection conditions safely under added restrictions [27, Appendix B]. Theorem 2 differs in two ways needed for an implementation rather than an idealized algorithm: it compares the *exact* three-output kernel with the *forced, fixed-point-truncated* two-output kernel actually executed by Algorithm 6, covering both the removed restart branch and the finite series depth T in one Rényi budget, and it makes the dependence on the key-norm window explicit through the gate triple (8)–(10), letting us pin a concrete $T = 33$ rather than appeal to the worst-case $\|\mathbf{v}\| = 1$ bound.

Security notion and the meaning of λ. Throughout this section, λ is the bit-security of the underlying search problem at the target level, i.e., the SIS-UF forgery problem of Gärtner's reduction [27]: exactly the quantity preserved by the Rényi-divergence arguments, which turn a search adversary against the implemented scheme into one against the ideal scheme with at most the stated bit loss. Like Dilithium [20, Table 1], we state security as UF-CMA at the target level and additionally report a strong-unforgeability (sUF) estimate in Table 2, which is lower because it must also rule out a fresh signature on an already-signed message; this does not change the UF-CMA target level used in Section 6.

Theorem 1 (Security of SampleZ)**.** *Let C_{ideal} be a cryptosystem that makes at most Q_Z independent queries to $\mathcal{D}_{\mathbb{Z},r,\mu}$, and suppose that it is λ-bit secure against a search problem. Let C_{real} be the same cryptosystem with each Gaussian query implemented by* SampleZ*, using at most Q_{bs} calls to* BaseSampler *and $Q_{\text{exp}} \geq Q_Z$ calls to* ApproxExp *over the lifetime of the attack. Write $\varepsilon_{\text{exp}}(x) = \mathsf{ApproxExp}(x) - \exp(x)$ and let $\eta = \max_{x<0} |\varepsilon_{\text{exp}}(x)/\exp(x)|$ be the acceptance-probability relative error. If*

$$R_{2\lambda-1}\big(\mathsf{BaseSampler}(r_s, U), \mathcal{D}_{\mathbb{Z}^+,r_s}\big) \leq 1 + \frac{1}{4Q_{\text{bs}}}, \quad (6)$$

and the resulting emitted-sample distortion satisfies

$$\frac{2\eta}{1-\eta} \leq \frac{1}{\sqrt{2(2\lambda-1)Q_{\text{exp}}}}, \quad (7)$$

then C_{real} is $(\lambda-2)$-bit secure against the same search problem.

Theorem 2 (Security of the forced $S_{\mathbf{v}}$ truncation)**.** *Consider the ideal IRS step, which evaluates the exact series $S_{\mathbf{v}}$ and outputs $\mathbf{y}-\mathbf{v}$, $\mathbf{y}+\mathbf{v}$, or the restart symbol $\perp$ with the probabilities f, g, $1-f-g$ of Equations (4) and (5), and the implemented step (Algorithm 6), which keeps the series terms indexed by $0,\dots,T-1$ and always outputs, assigning the residual restart mass to $\mathbf{y}+\mathbf{v}$. Here $\mathbf{y}$ denotes the step input, i.e., the current accumulator, written $\mathbf{z}$ in Algorithm 6. Write P for the ideal three-output kernel, $P^\star$ for the exact forced kernel with $P^\star(\mathbf{y}-\mathbf{v}) = f$ and $P^\star(\mathbf{y}+\mathbf{v}) = 1-f$, and $\widehat{P}^\star$ for the implemented kernel. Suppose the ideal scheme is λ-bit secure against a search problem with $\lambda \geq 64$, and the attack induces at most Q_{IRS} invocations of the IRS core. Since each invocation processes κ challenge positions, let $Q_{\text{step}} = \kappa Q_{\text{IRS}}$ be the total number of IRS steps, and assume the budget satisfies $Q_{\text{step}} \geq 2^{12}(2\lambda-1)$. Write $u = \langle \mathbf{y}, \mathbf{v}\rangle / r^2$ and $\gamma = \|\mathbf{v}\|^2/(2r^2)$, and fix a window bound $u_{\max}$ with $u_{\max} \geq 2+2\gamma$ for every admissible γ. Suppose that, for every secret accepted by key generation: (i) the window tail is negligible,*

$$Q_{\text{step}} \cdot 2e^{-u_{\max}^2/(4\gamma)} \leq 2^{-\lambda-6}; \quad (8)$$

(ii) every step satisfies $P(\perp) \leq \bar{\mu}$ with

$$Q_{\text{step}}\bar{\mu} \leq 2^{-32} \quad \text{and} \quad (2\lambda-2)\bar{\mu}Me^{u_{\max}+\gamma} \leq \frac{1-e^{-5\gamma}}{2}; \quad (9)$$

(iii) every step with $|u| \leq u_{\max}$ satisfies

$$\max_{o\in\{\mathbf{y}-\mathbf{v},\mathbf{y}+\mathbf{v}\}} \left| \frac{\widehat{P}^\star(o)}{P^\star(o)} - 1 \right| \leq \frac{1}{\sqrt{2(2\lambda-1)Q_{\text{step}}}}. \quad (10)$$

Then the implemented scheme is $(\lambda-1)$-bit secure. Moreover, the series truncation alone satisfies, for every input,

$$\max_{o\in\{\mathbf{y}-\mathbf{v},\mathbf{y}+\mathbf{v}\}} \left| \frac{P_T^\star(o)}{P^\star(o)} - 1 \right| \leq \frac{e^{-(T^2-2T-3)\gamma}}{1-e^{-\gamma}},$$

where $P_T^\star$ is the forced kernel computed from the truncated series $\widehat{S}_T$ in place of $S_{\mathbf{v}}$.

Composing the two theorems with Gärtner's reduction for the ideal scheme gives the end-to-end statement.

Corollary 1 (Security of implemented LITHIUM)**.** *Fix a security level of Table 2 and let λ be its target* SIS-UF *bit-security. Assume the ideal Gärtner scheme at this level is λ-bit UF-CMA secure under the NTWE and SIS assumptions [27], and assume the gates of Table 3 (instantiated in Appendix B.2). Then implemented* LITHIUM *(Algorithms 2 to 4, with the fixed-point* BaseSampler*,* ApproxExp*, and forced Algorithm 6) is $(\lambda-3)$-bit UF-CMA secure: at most two bits are lost by Theorem 1 and one by Theorem 2, and the norm and rANS gates contribute only the 2^{-100}-scale restart events.*

Table 3: High-probability bounds on function-call counts.

Item	LITHIUM-260	Budget
Q_{sig}	2^{64}	–
Q_{bsZ}	$2^{76.4}$	Eq. (6): $1+2^{-78.4}$
Q_{expZ}	$2^{76.4}$	Eq. (7): $2^{-43.3}$
Q_{step}	$2^{69.9}$	Eq. (10): $2^{-40.0}$

† Budgets use the LITHIUM-260 counts for all security levels because this level gives the largest call counts and the strictest gates. Here $Q_{\mathrm{sig}} = 2^{64}$ follows NIST's recommended signing-query bound, $Q_{\mathrm{bsZ}} = 2(1+\ell+m)nQ_{\mathrm{sig}}$, $Q_{\mathrm{expZ}} = Q_{\mathrm{bsZ}}$, and $Q_{\mathrm{step}} = \kappa Q_{\mathrm{sig}}$. The factor 2 for Q_{bsZ} covers its random loop count with overwhelming margin: with mean 1.25 trials per coefficient, a Chernoff bound gives lifetime overflow probability at most $2^{-3.0\cdot 10^{22}} \ll 2^{-260}$.

The call counts for the relevant implementation functions, together with the corresponding security budgets, are summarized in Table 3. We instantiate the table with the LITHIUM-260 counts because they give the largest query counts and therefore the strictest gates.

For the BaseSampler instances used inside SampleZ, we use RCDT tables scaled by 2^{93} and truncate the support at about ten standard deviations; the method satisfies the budget in Table 3. The SampleS RCDT parameters are generated by the same method and are reported, together with their Rényi divergences, in Appendix B.2.1.

For the sampler ApproxExp, Theorem 1 controls only the emitted-sample distribution, whose end-to-end relative error is at most $2^{-48.4}$ over the queried inputs, well below the $2^{-43.3}$ ApproxExp gate; Appendix B.2.2 derives this and addresses the larger reject-branch ratio, which Theorem 1 does not require. For IRS, we instantiate Theorem 2 with the window $u_{\max} = 8$, which coincides with the validated ApproxExp input domain: the tail gate (8) holds with lifetime mass below 2^{-651}, the forcing gate (9) holds with $\bar{\mu} \le 2^{-105}$, and the key-generation lower norm gate gives $\gamma_{\min} \approx 0.03$, under which the fixed-point step in Algorithm 6 with $T = 33$ has worst relative kernel deviation at most $2^{-43.2}$ over the whole window; Appendix B.2.3 derives the $T = 33$ choice and all three gate instantiations.

4.2 rANS Overflow

The signing output uses fixed-size signature slots even though rANS is internally variable length. We therefore reserve enough space for the rANS streams so that overflow occurs with probability about 2^{-101}, and treat overflow as a signing retry event.

The concrete byte layout, entropy model, and signature-size formula are given in Appendix B.2.4; the same derivation sizes the rANS reserve by a Bernstein tail bound and corrects Gärtner's signature-size estimate [27, Appendix A.2], which models the MakeHint output as Gaussian rather than as the bucket-crossing distribution induced by the rounding buckets.

5 Optimization and Implementation

As in other lattice signatures based on discrete Gaussian sampling, including Falcon [25] and HAETAE [12], the signing bottleneck in LITHIUM is Gaussian mask generation, namely SampleY. The new IRS component is also computationally intensive because each nonzero challenge coefficient triggers a threshold decision based on the IRS series. This section describes the algorithmic and implementation optimizations that make these two components practical while preserving the security budgets from Section 4.

5.1 Vectorized Gaussian Sampling

SampleY mainly relies on SampleZ, the $k = 128$ instantiation of the convolution sampler of Section 2.4: BaseSampler scans a 93-bit reverse-CDT table for standard deviation $r/128$, and the candidate $z = 128x + y$ is accepted with probability

$$\exp\bigl(-y(y+256x)/(2r^2)\bigr), \tag{11}$$

so the hot inner-loop components are BaseSampler and the ApproxExp used for computing $\exp(\cdot)$.

5.1.1 Vectorized BaseSampler

We adopt the vectorized technique for AVX2 and AVX-512 from [50]: expose independent lanes with a batched sampler, and accelerate the core table scan with vector instructions. In LITHIUM, SampleY draws $(1+\ell+m)n$ independent Gaussian samples per signing attempt, so the half-Gaussian calls naturally form a batched interface that runs the BaseSampler table scan on several independent random words at once and returns one counter per lane.

The core operation is a constant-time comparison between a 93-bit random word and each RCDT entry. We store each value as three 31-bit limbs. Each limb fits in one 32-bit SIMD lane, so an AVX2 register processes eight limbs and an AVX-512 register processes sixteen limbs. The implementation then realizes the 93-bit comparison by a low-to-high limb subtraction with borrow propagation, or equivalently by a low-to-high limb comparison, and accumulates the comparison masks into the output counters.

5.1.2 ApproxExp

According to [50, Sec. 9], ApproxExp-related computations can account for up to 36% of Falcon signing time. Falcon

Algorithm 5: ApproxExp(a_{60})

Input: Q60 input $a_{60} \approx a2^{60}$; $B = 2^{\beta} = 8$; RCP $\approx (B/\ln 2)2^{57}$; LB $\approx (\ln 2/B)2^{64+\beta}$; Q63 coefficients $C_0, \ldots, C_6$ and table $\mathsf{Tab}[j] \approx 2^{-j/B}2^{63}$

Output: Q63 value round$(e^{-a}2^{63})$

$V \leftarrow \mathsf{mulh64}(a_{60}, \mathsf{RCP})$ // $(Ba/\ln 2)2^{53}$

$m \leftarrow ((V + 2^{53} - 1) \gg 53) + (B-1)$ // ceil

$h \leftarrow m \gg \beta$; $j \leftarrow m \,\&\, (B-1)$ // $m = Bh + j$

$R \leftarrow m \cdot \mathsf{LB} - (a_{60} \ll (\beta+4))$ // $r_\ell 2^{64+\beta}$

$X \leftarrow R \gg \beta$ // Q64 residual

$p \leftarrow C_6$

for $i \leftarrow 5, \ldots, 0$ **do**

  $p \leftarrow \mathsf{mulh64}(p, X) + C_i$ // Horner

$s \leftarrow \mathsf{CTLookup}(\mathsf{Tab}, j)$ // full scan

$W \leftarrow \mathsf{mulh64}(p, s) \ll 1$ // Q63 product

return $\lfloor (W + 2^{h-1})/2^h \rfloor$ // round-to-nearest

approximates exp on $[0, \ln 2]$ with a degree-12 polynomial[2] which is evaluated by Horner's method. This becomes a bottleneck because Horner evaluation is serial: each multiplication waits for the previous multiplication result, creating a long dependency chain.

The approximation target can be viewed through the standard Chebyshev minimax bound. For a smooth function f on the interval $[a,b]$,

$$E_d(f;[a,b]) \le \frac{(b-a)^{d+1}}{2^{2d+1}(d+1)!} \cdot \max_{\xi \in [a,b]} |f^{(d+1)}(\xi)|. \qquad (12)$$

Here E_d denotes the best uniform error achievable by a degree-d real polynomial. The bound shows that, for a fixed error budget, narrowing the approximation interval can reduce the required degree. A second issue comes from Equation (7): the term $|\varepsilon_{\exp}(x)/(1-\exp(x))|$ becomes large as x approaches 0, so we prefer the polynomial interval to stay away from 0.

Our optimized algorithm. For SampleZ, we need e^{-a} for $a \in [0, 7.2)$. Let $B = 2^{\beta}$ be the number of table segments and set

$$m = \left\lceil \frac{Ba}{\ln 2} \right\rceil + B - 1, \qquad m = Bh + j, \quad 0 \le j < B.$$

Define $r_\ell = m \ln 2/B - a$. Since $\lceil Ba/\ln 2 \rceil - Ba/\ln 2 \in [0,1)$, we have $r_\ell \in [(B-1)\ln 2/B, \ln 2)$. Therefore

$$e^{-a} = 2^{-h} \cdot 2^{-j/B} \cdot e^{r_\ell}.$$

This range reduction moves the polynomial approximation to a narrow interval away from 0: $[(B-1)\ln 2/B, \ln 2)$. The factor 2^{-h} is a shift, $2^{-j/B}$ is a constant-time table lookup, and e^{r_ℓ} is approximated by a degree-6 integer polynomial generated with the GALACTICS toolkit [6].

Larger B narrows the interval and can lower the polynomial degree, but it also increases the full-table-scan cost. We found $B = 8$ to be the best point for our implementation. Algorithm 5 gives the resulting ApproxExp routine: the input is scaled by 2^{60} and the output by 2^{63}. In the algorithm, $\mathsf{mulh64}(x,y)$ returns the high 64 bits of the product xy, and CTLookup scans the full table and selects the requested entry by masking. Its fixed-point error analysis is given in Appendix B.2.2.

Microarchitectural optimization. On modern CPUs, the latency of a 64-bit multiplication is typically at least three cycles [1]. The dependency chain in Horner evaluation prevents these dense multiplications from fully using the backend pipeline. As with the batching strategy in Section 5.1.1, we therefore execute four independent ApproxExp instances together. This amortizes the full-table scan and exposes more instruction-level parallelism, especially for the multiplication-heavy Horner chains.

Faster opt-in variants. AVX2 lacks a native four-lane 64×64 high-product, so our opt-in AVX2 version uses four-lane double-precision floating-point multiplication with a Sollya-generated real-coefficient polynomial [13], and our AVX-512IFMA version uses the eight-lane 52-bit integer fused multiply-add with the same range reduction; both trade byte-for-byte KAT compatibility (and, for the floating-point path, side-channel hygiene) for speed, while staying verifier-interoperable, since verification depends only on the public relation and encoding. Our $4\times$ integer method is $1.7\times$ faster than Falcon's degree-12 polynomial, and the AVX2 and AVX-512IFMA opt-in variants reach $8.8\times$ and $10.5\times$; Appendix B.2.2 tabulates the per-call cycle counts.

5.2 Iterative Rejection Decisions

An IRS step means one iteration of the κ-round loop in Algorithm 1: given the current accumulator $\mathbf{z}$ and the shifted secret vector $\mathbf{v} = x^j\mathbf{s}$ for one challenge monomial, the implementation must choose the next accumulator $\mathbf{z} - \mathbf{v}$ or $\mathbf{z} + \mathbf{v}$. The exact IRS rule also has an abort branch, but our parameters make the total IRS abort probability below 2^{-101}. The implementation therefore uses the forced-output rule in Algorithm 6, which assigns the residual abort mass to the plus branch; Theorem 2 proves that this forced step, including the series truncation, loses at most one bit of search security, with the gates instantiated in Appendix B.2.3.

In Algorithm 6, $\mathsf{Horner}_T(C,x)$ denotes the T-term Horner evaluation of $\sum_{k=0}^{T-1} c_k x^k$ for the coefficient table $C = (c_0, \ldots, c_{T-1})$, and $\mathsf{Select}(m,a,b)$ denotes a constant-time mask selection that returns a when the bit m is set and b otherwise.

The exact IRS step evaluates the probabilities $f_{\mathbf{v}}(\mathbf{z})$ and $g_{\mathbf{v}}(\mathbf{z})$ through the alternating series $S_{\mathbf{v}}(\mathbf{z})$ of Equation (3).

[2] The expm_p63 subroutine at https://github.com/pornin/c-fn-dsa/blob/main/sign_sampler.c (commit: 3aa5048)

Algorithm 6: IRSStep$(\mathbf{z},\mathbf{v},\gamma,C,r_{63})$

Input: current accumulator $\mathbf{z}$, shifted secret $\mathbf{v}=x^j\mathbf{s}$, $\gamma=\|\mathbf{v}\|^2/(2r^2)$, table $C=(c_0,\dots,c_{T-1})$ with $c_k=(-1)^k e^{-k^2\gamma}$, and random word $r_{63}\in\{0,\dots,2^{63}-1\}$

Output: updated accumulator $\mathbf{z}'\in\{\mathbf{z}-\mathbf{v},\mathbf{z}+\mathbf{v}\}$

$u\leftarrow\langle\mathbf{z},\mathbf{v}\rangle/r^2$; $a\leftarrow|u|$; $a_c\leftarrow\min(a,2\gamma)$

$w\leftarrow e^{-a}$; $w'\leftarrow e^{a_c}$ // implemented by ApproxExp-style fixed-point routines

$S_+\leftarrow\mathsf{Horner}_T(C,w)$; $S_-\leftarrow\mathsf{Horner}_T(C,w')$ // $S_\gamma(|u|)$ and opposite direction

$(S_+,S_-)\leftarrow\mathsf{Select}(u<0,(S_-,S_+),(S_+,S_-))$ // values for $+u$ and $-u$

$\theta_-\leftarrow\mathsf{Select}(u\geq 2\gamma,S_+,1-S_-)$ // $M\cdot f(u)$, the minus-branch threshold

$R\leftarrow r_{63}M/2^{63}$

if $R<\theta_-$ **then return** $\mathbf{z}-\mathbf{v}$

return $\mathbf{z}+\mathbf{v}$ // forced output: residual abort mass goes to plus

Writing $u=\langle\mathbf{z},\mathbf{v}\rangle/r^2$ and $\gamma=\|\mathbf{v}\|^2/(2r^2)$, each series ratio becomes $\exp(-ku-k^2\gamma)$, so Equations (4) and (5) reduce to the one-dimensional rule evaluated on $S_\gamma(u)=\sum_{k\geq 0}(-1)^k e^{-ku-k^2\gamma}$; Appendix A gives the full reduction and the resulting closed form for $f(u)$ and $g(u)$.

A direct implementation would repeatedly evaluate exponentials while walking this sum for every challenge position. We instead separate the per-key and per-step quantities. For a fixed key, γ is fixed for all shifted secret vectors $\mathbf{v}=x^j\mathbf{s}$, so the coefficients

$$c_k=(-1)^k e^{-k^2\gamma},\qquad 0\leq k<T,$$

are precomputed once per signature and stored in the table C used by Algorithm 6. A truncated call to $S_{\mathbf{v}}(\mathbf{z})$ is then a Horner evaluation of $\sum_{k=0}^{T-1}c_k x^k$ with $x=e^{-u}$, rather than a sequence of separate exponentials.

The rule above needs the two directions u and $-u$. The direction with larger inner product uses the stable input $x=e^{-|u|}$. The other direction is needed only in the middle region $|u|<2\gamma$, so we evaluate it as $x=e^{a_c}$ with $a_c=\min(|u|,2\gamma)$ and use masked selection to assign the two values to the plus or minus branch according to the sign of u. The random word r_{63} in Algorithm 6 is the fixed-point Bernoulli threshold used to choose between the two forced outputs. The per-step work becomes two exponentials, two Horner chains, and mask-based selection, plus the amortized coefficient precomputation.

The truncation depth is $T=33$. The executable reference checks the fixed-point implementation against the real-arithmetic infinite series at the key-norm lower endpoint, at every integer inner product of the proof window $|u|\leq 8$. The worst relative kernel deviation is about 2^{-43} for all security levels, below the approximately 2^{-40} Rényi gate instantiated in Appendix B.2.3. The decision code implements the sign, absolute value, region selection, and final add/subtract with masks or conditional moves, and replaces secret-derived divisions by reciprocal multiplications.

5.3 NTT

ML-KEM uses $q=3329$, and its NTT implementation can fully exploit AVX2's sixteen-lane 16-bit mode with Seiler's signed Montgomery reduction [46]. Our moduli are larger: $q=57601,30977,64513$ for the three security levels. Although each modulus still fits in 16 bits, there are not enough redundant bits to use the same signed lazy-reduction strategy safely.

We therefore use unsigned Montgomery reduction. Every addition and subtraction must handle carries and borrows in constant time, which makes our AVX2 NTT slower than ML-KEM's sixteen-lane 16-bit NTT. It is still faster than ML-DSA's eight-lane 32-bit-modulus NTT. We also implement the NTT arithmetic with AVX-512, which widens the same unsigned-reduction strategy.

5.4 Other Components

rANS. Our rANS implementation follows Duda's method [23]. As discussed in Section 4.2, the reserve is sized so that overflow occurs with probability about 2^{-101}.

We also vectorize several other subroutines with AVX2 and AVX-512, including ExpandA, SampleC, MakeHint, and pkDecode.

5.5 KAT Compatibility

As discussed in Section 1, Falcon did not design its Gaussian sampler with KAT-compatible vectorization in mind. Later vectorized samplers [50] remain sign-verify interoperable with the original algorithm, but they cannot reproduce the same KATs because their SHAKE-based pseudorandom stream is generated and consumed in a different pattern.

LITHIUM makes this boundary explicit in the implementation design. The reference, AVX2, and AVX-512 backends use the same pseudorandom-generation and consumption pattern for KAT-critical computations, including Gaussian sampling and byte-level encoding. Concretely, every XOF expansion (ExpandA, SampleS, SampleY) is structured as exactly eight independent streams $\rho\|b$ for $b=0,\dots,7$ from the outset. This number is dictated by the widest backend: an AVX-512 Keccak permutation processes eight 1600-bit sponges in parallel (Keccak8x), so eight streams map one-to-one onto its lanes, the AVX2 backend runs them as two Keccak4x calls, and the scalar reference runs eight sequential sponges—all consuming the same bytes in the same order. A four-stream

Table 4: Verification-key and signature sizes at different security levels.

Scheme	Sec.	vk	sig	vk+sig
LITHIUM-120	125	1056	1187	2243
ML-DSA-44 [36]	123	1312	2420	3732
HAETAE-120 [12]	119	992	1474	2466
Falcon-512 [25]	120	897	666	1563
HAWK-512 [10]	120	1024	555	1579
Raccoon-128 [15]	134	2256	11524	13780
NTRU+Sign-512 [47]	93	768	751	1519
LITHIUM-180	189	1472	1618	3090
ML-DSA-65 [36]	183	1952	3309	5261
HAETAE-180 [12]	180	1472	2349	3821
Raccoon-192 [15]	193	3160	14544	17704
NTRU+Sign-1024 [47]	211	1664	1551	3215
LITHIUM-260	268	2080	2348	4428
ML-DSA-87 [36]	252	2592	4627	7219
HAETAE-260 [12]	256	2080	2948	5028
Falcon-1024 [25]	273	1793	1280	3073
HAWK-1024 [10]	260	2440	1221	3661
Raccoon-256 [15]	284	4064	20330	24394

* Security is in classical bits; sizes are in bytes. Security level presented for HAWK is based directly on target NIST security category.

layout would either idle half of the AVX-512 lanes or force the wide backend to interleave the XOF output differently from the scalar code, which is exactly the mismatch that breaks byte-for-byte KATs. As a result, the three default backends are KAT-compatible, while optional fast ApproxExp variants using AVX2 and AVX-512IFMA are kept as separate performance configurations when they intentionally trade byte-for-byte KAT identity for speed.

6 Evaluation

Gärtner's theoretical construction shows that IRS-based lattice signatures can be compact [27]. Our evaluation asks the corresponding implementation questions: is LITHIUM competitive in signing speed after the optimizations of Section 5, and can existing testing and audit tools support the constant-time claim of Section 1?

Table 5: Optimized-implementation performance.

Scheme	Impl.	KeyGen	Sign	Tail	Verify
LITHIUM-120	AVX2	264k	337k	1.05	65k
LITHIUM-180	AVX2	360k	547k	1.03	98k
LITHIUM-260	AVX2	539k	689k	1.02	118k
LITHIUM-120	AVX2Exp	262k	303k	1.03	65k
LITHIUM-180	AVX2Exp	362k	494k	1.02	98k
LITHIUM-260	AVX2Exp	548k	639k	1.04	117k
LITHIUM-120	AVX-512	172k	211k	1.04	57k
LITHIUM-180	AVX-512	241k	344k	1.03	83k
LITHIUM-260	AVX-512	363k	442k	1.03	103k
LITHIUM-120	IFMAExp	173k	166k	1.04	57k
LITHIUM-180	IFMAExp	243k	284k	1.03	83k
LITHIUM-260	IFMAExp	368k	361k	1.04	103k
ML-DSA-44	AVX2	80k	191k	4.08	86k
ML-DSA-65	AVX2	137k	305k	3.98	139k
ML-DSA-87	AVX2	215k	388k	3.31	217k
HAETAE-120	AVX2	420k	1249k	6.08	92k
HAETAE-180	AVX2	615k	1443k	4.90	147k
HAETAE-260	AVX2	680k	1959k	6.02	173k
HAWK-512	AVX2	5105k	94k	1.06	122k
HAWK-1024	AVX2	29031k	189k	1.03	253k
Falcon-512	SSE2	10483k	824k	1.03	47k
Falcon-1024	SSE2	38543k	1663k	1.02	89k
Falcon-512	AVX2	–	688k	1.03	–
Falcon-1024	AVX2	–	1394k	1.03	–
Falcon-512	AVX-512	–	645k	1.03	–
Falcon-1024	AVX-512	–	1312k	1.02	–

* The suffix k denotes thousands of CPU cycles. KeyGen and Sign are StQ_2; Tail is p99/p50; Verify is the median. Dashes: not reported upstream. Falcon SSE2: https://github.com/pornin/c-fn-dsa@3aa5048; Falcon AVX: https://github.com/Ji-Peng/VecFalcon@61b326e from [50].

6.1 Benchmark Configurations

All benchmarks run on an Intel Core i7-11700K at 3.60GHz, which supports both AVX2 and AVX-512, running Ubuntu 24.04 with GCC 13.3; we disable Turbo Boost and hyper-threading to reduce variance. LITHIUM is compiled with '-O3'; for other schemes, we use each project's default compilation options, since '-O2' vs. '-O3' makes little difference once the hot loops are already vectorized with intrinsics.

Signing metric. As recommended by Bernstein [7], rejection sampling can make the median unstable, so we report the stabilized median StQ_2, defined as the average over the $3/8$–$5/8$ quantile range across 10000 runs, for both key generation and signing. Because the community has paid attention to ML-DSA's long signing-time tail [39, 45], we also report

tail = p99/p50.

Key-generation and verification metrics. For verification, we report the median over 10000 runs.

6.2 Size and Speed

What are fair comparison targets? ML-DSA and HAETAE are the closest baselines, since both follow the lattice-based Fiat-Shamir-with-aborts paradigm; Falcon is the next most relevant standardized target, following a hash-and-sign paradigm whose signing core also relies on discrete Gaussian sampling. We report additional schemes for broader context.

Table 4 reports the size comparison. Across the three security levels, LITHIUM signatures are about half the size of the corresponding ML-DSA signatures, and the verification-key-plus-signature footprint is about 60% of ML-DSA. Compared with HAETAE, LITHIUM reduces the verification-key-plus-signature footprint by 223, 731, and 600 bytes at the three security levels, respectively.

What the size gap means in deployment. The 1,233-byte per-signature reduction over ML-DSA-44 is a first-order saving where post-quantum signatures hurt most: a TLS 1.3 handshake carrying several signatures saves 3–4 KB, and a Certificate Transparency or signed-log archive roughly halves its dominant storage term. The effect is sharpest in DNSSEC, where a response should fit the de-facto 1,232-byte EDNS(0) buffer: ML-DSA-44's 2,420-byte signature alone is roughly twice that, whereas LITHIUM-120's 1,187 bytes make one-datagram responses plausible for small records.

Table 5 reports the optimized-implementation numbers (StQ_2 for all key-generation and signing rows; Appendix B.2.6 gives the matching reference-implementation table). All LITHIUM backends share the IRS optimization of Section 5.2, and the AVX2/AVX-512 backends add the vectorized sampling, NTT, and other components of Section 5. AVX2Exp and IFMAExp replace the integer ApproxExp with the floating-point AVX2 and 52-bit AVX-512IFMA versions of Section 5.1.2.

LITHIUM signs faster than HAETAE at every optimization level and faster than ML-DSA in the reference implementation (Appendix B.2.6); we omit the AVX-512 Dilithium of Xu et al. [48] as it is not interoperable with ML-DSA. For AVX2, ML-DSA signs about 1.8× faster in StQ_2 but with a severe long tail (p99 is 3–4× p50), whereas LITHIUM's AVX2 tail stays near one and its signer is 2.6–3.7× faster than HAETAE. Our fastest configuration (AVX-512IFMA ApproxExp) signs 7–15% faster than ML-DSA AVX2, 5.1–7.5× faster than HAETAE AVX2, and 3.6–3.9× faster than Falcon.

Cross-platform check on AMD Zen 5. We also ran the complete optimized- and reference-implementation benchmark suite on a Google Cloud C4D instance (AMD EPYC 9B45, Zen 5); Appendix B.2.7 gives the full tables. The picture holds and sharpens: LITHIUM-120's fastest signer is 1.1–1.2× faster than ML-DSA and 5.7–8.6× faster than HAETAE across the three levels with the same near-one tail, and the KAT-locked AVX-512 default matches ML-DSA AVX2.

6.3 Profiling

Profiling the AVX2 LITHIUM-120 signer confirms where the cost is: SampleY dominates at 64.7% of signing time (Keccak 34.1%, ApproxExp 15.9%), which is why Sections 5.1 and 5.1.2 matter end to end. The IRS decision logic is second at 18.8%, so the table-assisted Horner rewrite of Section 5.2 also helps.

6.4 Secure-Implementation Evidence

We evaluate the secure-implementation claim from Section 1: LITHIUM targets not whole-program fixed-time execution or physical side-channel resistance, but a signing core that avoids secret-dependent control flow, memory addresses, division, square roots, and gathers. We statically audit the generated assembly across all backends, security levels, GCC and Clang, $-O3$ and $-Os$, and the ApproxExp variants for secret-dependent branches, addressing, and high-risk instructions; Appendix B.2.8 gives the full audit methodology, and the only occurrences found are public SHAKE lane operations or public-length divisions.

We complement this with statistical and dynamic tests. End-to-end dudect [44] experiments (200,000 traces per backend/level, fixed- vs. random-secret) give a largest Welch statistic $|t| = 1.23$, well below the $|t| < 4.5$ threshold. TIMECOP [9] reports zero variable-latency hits for the reference and AVX2 signers (AVX-512 is unsupported, so its evidence is the assembly audit and dudect). Finally, motivated by recent ML-DSA bug analyses [8], KAT, differential-fuzzing, and fault-injection tests check that optimization preserves transcript structure, randomness use, and rejection behavior.

7 Related Work

Fiat-Shamir lattice signatures. Fiat-Shamir with aborts turns lattice identification protocols into signatures while hiding the secret-dependent part of the response through rejection sampling [33, 34], leading to practical schemes such as Dilithium [20], standardized as ML-DSA [36]. Devevey, Fawzi, Passelègue, and Stehlé characterized this template's rejection-sampling cost and how close it can come to optimal [17]. Gärtner's iterative rejection sampling (IRS) construction takes a different route: rather than one global abort decision, it builds the response through a sequence of local rejection-sampling steps, yielding much more compact parameter sets [27].

Compact Fiat-Shamir variants. BLISS showed that changing the rejection-sampling distribution, in particular through bimodal Gaussians, can substantially reduce signature size [19], also making Gaussian-sampler implementability a central issue. HAETAE uses hyperball bimodal rejection sampling [12] and was selected in the KpqC competition [31]; NTRU+Sign follows a modern NTRU-based bimodal design [47]; and G+G replaces aborts by convolved Gaussians [18]. LITHIUM differs by implementing IRS, budgeting its approximation errors, and introducing optimizations that make its performance practical.

Hash-and-sign signatures. The compactness target outside Fiat-Shamir signatures is hash-and-sign. The Nguyen–Regev attack on GGH and early NTRU signatures showed why the output distribution must hide the signing basis [38], and GPV gave the modern lattice hash-and-sign template [28]. Falcon instantiates this with NTRU lattices and fast Fourier sampling [25]; Espitau, Fouque, Gérard, et al. later gave Mitaka, a simpler, maskable variant [24]; and HAWK is another compact alternative with a different assumption and profile [21]. LITHIUM is not a replacement for Falcon or HAWK; it asks whether Fiat-Shamir signatures can move closer to their compactness while keeping a deployment profile closer to ML-DSA.

Secure and optimized implementations. Lattice signatures make implementation choices part of the security argument: Groot Bruinderink, Hülsing, Lange, and Yarom cache-attacked a non-constant-time BLISS Gaussian sampler [11], and Pessl, Groot Bruinderink, and Yarom broke a deployed BLISS-B implementation [41]. GALACTICS treated Gaussian sampling, ApproxExp, and rejection sampling as constant-time problems rather than black-box subroutines [6]; even then, Marzougui, Wisiol, Gersch, et al. later reported ML side-channel attacks against it [35], which is why we keep physical side channels out of scope (Section 1). Prest's Rényi-divergence bounds [42], the isochronous-sampling framework [30], and samplers such as FACCT [51] compose and isochronize sampler approximations, and vectorized Falcon signing showed such samplers benefit from AVX2/AVX-512/NEON/RVV at the cost of transcript compatibility [50]. LITHIUM follows this line: its approximation budgets, sampler design, IRS decision code, vector backends, and KAT boundary are specified together, not as post-hoc optimizations.

8 Limitations and Ethics

Underlying hardness assumption. Following Gärtner's IRS construction [27], LITHIUM relies on NTWE, a combination of the NTRU and LWE problems [26, 27] related to both rank-k Module-LWE and NTRU-style hardness; Appendix C expands the argument and addresses the maturity concern. The implementation techniques here are not NTWE-specific and remain applicable if future work replaces NTWE with Module-LWE.

Ethics and artifact handling. This work studies a compact lattice signature's design and implementation; it does not measure live systems, collect human-subject data, or disclose deployed vulnerabilities (Appendices C and D).

Generative AI disclosure. Generative AI assisted with editorial polishing and test-code drafting; all outputs were inspected and validated by the authors.

9 Conclusion

LITHIUM demonstrates that iterative rejection sampling can support a practical Fiat-Shamir lattice signature rather than only a compact asymptotic construction. The central lesson is that compactness and speed must be engineered together: the parameter search, IRS truncation budget, Gaussian sampler, ApproxExp evaluation, rANS encoding, and vectorized backends all affect whether the theoretical size advantage survives in software. With these choices, LITHIUM reaches roughly half the signature size of ML-DSA across the three security levels, while its fastest signer is slightly faster than the optimized ML-DSA baseline and substantially faster than HAETAE and Falcon on the measured platform. The implementation evidence further shows that these gains can coexist with backend interoperability and a disciplined signing-secret-dependent constant-time core. More broadly, the optimization techniques developed here are not tied to a single parameter set: they provide a reusable path for future lattice signatures that combine iterative rejection sampling, Gaussian sampling, and high-throughput vector software.

References

[1] Andreas Abel and Jan Reineke. uops.info: Characterizing latency, throughput, and port usage of instructions on Intel microarchitectures. In *ASPLOS '19*, ASPLOS '19, pages 673–686, New York, NY, USA, 2019. ACM.

[2] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 99–108. ACM, 1996.

[3] Martin R. Albrecht and contributors. lattice-estimator: Security estimates for lattice problems. Github repository, 2021. Accessed: 2026-06-01.

[4] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.

[5] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new

hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16*, pages 327–343. USENIX Association, 2016.

[6] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Mélissa Rossi, and Mehdi Tibouchi. GALACTICS: Gaussian sampling for lattice-based constant- time implementation of cryptographic signatures, revisited. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019*, pages 2147–2164. ACM, 2019.

[7] Daniel J. Bernstein. Robust statistics for rejection-sampling timings. Author-hosted preprint, 2025. Accessed 2026-06-06.

[8] Daniel J. Bernstein. Exploiting ML-DSA bugs. Author-hosted preprint, 2026. Accessed 2026-06-06.

[9] Daniel J. Bernstein, Karthikeyan Bhargavan, Shivam Bhasin, Anupam Chattopadhyay, Tee Kiah Chia, Matthias J. Kannwischer, Franziskus Kiefer, Thales B. Paiva, Prasanna Ravi, and Goutam Tamvada. KyberSlash: Exploiting secret-dependent division timings in Kyber implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(2):209–234, 2025.

[10] Joppe W. Bos, Olivier Bronchain, Léo Ducas, Serge Fehr, Yu-Hsuan Huang, Thomas Pornin, Eamonn W. Postlethwaite, Thomas Prest, Ludo N. Pulles, and Wessel van Woerden. HAWK version 1.0.2. https://hawk-sign.info, sep 2024.

[11] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016*, pages 323–345. Springer, 2016.

[12] Jung Hee Cheon, Hyeongmin Choe, Julien Devevey, Tim Güneysu, Dongyeon Hong, Markus Krausz, Georg Land, Marc Möller, Damien Stehlé, and MinJune Yi. HAETAE: shorter lattice-based fiat-shamir signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(3):25–75, 2024.

[13] Sylvain Chevillard, Mioara Joldeş, and Christoph Quirin Lauter. Sollya: An environment for the development of numerical codes. In *Mathematical Software - ICMS 2010 - Third International Congress on Mathematical Software*, volume 6327, pages 28–31. Springer, 2010.

[14] Lynn Chua, Hao Chen, Yongsoo Song, and Kristin Lauter. On the concrete security of LWE with small secret. *La Matematica*, 3(3):1032–1068, June 2024.

[15] Rafael del Pino, Thomas Espitau, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, Mélissa Rossi, and Markku-Juhani O. Saarinen. Raccoon: A side-channel secure signature scheme. NIST Additional Digital Signature Schemes Round 1 submission, 2023. Accessed: 2026-06-06.

[16] Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. Raccoon: A masking-friendly signature proven in the probing model. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024*, volume 14920, pages 409–444. Springer, 2024.

[17] Julien Devevey, Omar Fawzi, Alain Passelègue, and Damien Stehlé. On rejection sampling in lyubashevsky's signature scheme. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022*, pages 34–64. Springer, 2022.

[18] Julien Devevey, Alain Passelègue, and Damien Stehlé. G+G: A Fiat–Shamir lattice signature based on convolved Gaussians. In *Advances in Cryptology – ASIACRYPT 2023*, volume 14444, pages 37–64. Springer, 2023. Full version: Cryptology ePrint Archive, Paper 2023/1477.

[19] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013*, volume 8042, pages 40–56. Springer, 2013.

[20] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2018(1):238–268, Feb. 2018.

[21] Léo Ducas, Eamonn W. Postlethwaite, Ludo N. Pulles, and Wessel P. J. van Woerden. Hawk: Module LIP makes lattice signatures fast, compact and simple. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022*, pages 65–94. Springer, 2022.

[22] Léo Ducas and Ludo N. Pulles. Does the dual-sieve attack on learning with errors even work? In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023*, pages 37–69. Springer, 2023.

[23] Jarek Duda. Asymmetric numeral systems: Entropy coding combining speed of Huffman coding with compression rate of arithmetic coding, 2013.

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

[25] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. NIST Post-Quantum Cryptography Project, Round 3 submission, 2020. Specification version 1.2, dated 2020-10-01; project site https://falcon-sign.info/.

[26] Joel Gärtner. NTWE: A natural combination of NTRU and LWE. In *Post-Quantum Cryptography - 14th International Workshop, PQCrypto 2023*, pages 321–353. Springer, 2023.

[27] Joel Gärtner. Compact lattice signatures via iterative rejection sampling. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025 - 45th Annual International Cryptology Conference*, pages 514–547. Springer, 2025.

[28] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM, 2008.

[29] Jianhua Hou and Haodong Jiang. Careful with the ring: Enhanced hybrid decoding attacks against module/ring-LWE. Cryptology ePrint Archive, Paper 2026/366, 2026.

[30] James Howe, Thomas Prest, Thomas Ricosset, and Mélissa Rossi. Isochronous Gaussian Sampling: From Inception to Implementation. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, volume 12100, pages 53–71. Springer, 2020.

[31] KpqC Competition. KpqC competition final selected algorithms. Official competition announcement, 2025. Accessed 2026-06-11.

[32] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, 2015.

[33] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *Advances in Cryptology - ASIACRYPT 2009*, volume 5912, pages 598–616. Springer, 2009.

[34] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012*, volume 7237, pages 738–755. Springer, 2012.

[35] Soundes Marzougui, Nils Wisiol, Patrick Gersch, Juliane Krämer, and Jean-Pierre Seifert. Machine-learning side-channel attacks on the GALACTICS constant-time implementation of BLISS. In *ARES 2022: The 17th International Conference on Availability, Reliability and Security*, pages 34:1–34:11. ACM, 2022.

[36] National Institute of Standards and Technology. Module-lattice-based digital signature standard. FIPS 204, National Institute of Standards and Technology, August 2024.

[37] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. FIPS 203, National Institute of Standards and Technology, August 2024.

[38] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of ggh and ntru signatures. In *Advances in Cryptology - EUROCRYPT 2006*, page 271–288. Springer Berlin Heidelberg, 2006.

[39] NIST PQC Forum. Announcement of the 3rd round onramp candidates, discussion thread and replies, 2026. The thread focuses on the tail-latency issue in ML-DSA signing. Accessed: 2026-06-06.

[40] Niklas Nolte, Mohamed Malhou, Emily Wenger, Samuel Stevens, Cathy Yuanchen Li, François Charton, and Kristin E. Lauter. The cool and the cruel: Separating hard parts of LWE secrets. In *Progress in Cryptology - AFRICACRYPT 2024*, Lecture Notes in Computer Science, pages 428–453. Springer, 2024.

[41] Peter Pessl, Leon Groot Bruinderink, and Yuval Yarom. To BLISS-B or not to be: Attacking strongSwan's implementation of post-quantum signatures. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017*, pages 1843–1855. ACM, 2017.

[42] Thomas Prest. Sharper bounds in lattice-based cryptography using the Rényi divergence. In *Advances in Cryptology - ASIACRYPT 2017*, volume 10624, pages 347–374. Springer, 2017.

[43] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 84–93. ACM, 2005.

[44] Oscar Reparaz, Josep Balasch, and Ingrid Verbauwhede. Dude, is my code constant time? In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017*, pages 1697–1702. IEEE, 2017.

[45] Sébastien Riou, Jong-Yeon Park, Liga Anwar, Axel Poschmann, and Michael Hutter. Apples, oranges, and signatures: Pitfalls and methodology in ML-DSA benchmarking. Slides presented at MAgiCS, 2026. Accessed 2026-06-06.

[46] Gregor Seiler. Faster AVX2 optimized NTT multiplication for ring-LWE lattice cryptography. *IACR Cryptology ePrint Archive*, 2018:39, 2018.

[47] Joo Woo, Jonghyun Kim, Ga Hee Hong, Seungwoo Lee, Minkyu Kim, Hochang Lee, and Jong Hwan Park. Ntru+sign: compact ntru-based signatures using bimodal distributions. *Des. Codes Cryptogr.*, 94(1):2, 2026. Preprint: Cryptology ePrint Archive, Paper 2025/106.

[48] Runqing Xu, Debiao He, Min Luo, Cong Peng, and Xiangyong Zeng. Optimizing Dilithium implementation with AVX2/AVX-512. *ACM Transactions on Embedded Computing Systems*, 23(6), September 2024.

[49] Jipeng Zhang, Junhao Huang, Lirui Zhao, Donglong Chen, and Çetin Kaya Koç. ENG25519: Faster TLS 1.3 handshake using optimized X25519 and Ed25519. In Davide Balzarotti and Wenyuan Xu, editors, *33rd USENIX Security Symposium, USENIX Security 2024*. USENIX Association, 2024.

[50] Jipeng Zhang and Jiaheng Zhang. Vectorized Falcon-Sign implementations using SSE2, AVX2, AVX-512F, NEON, and RVV. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2026:302–324, 2026.

[51] Raymond K. Zhao, Ron Steinfeld, and Amin Sakzad. FACCT: FAst, Compact, and Constant-Time Discrete Gaussian Sampler over Integers. *IEEE Transactions on Computers*, 69(1):126–137, 2020.

A Algorithms and Encoding Details

This appendix fixes the byte-level choices for reproducible KATs.

NTT arithmetic. We use NTT for coefficient-wise products in T_q and omit standard butterfly schedules and SIMD layouts.

Seed expansion and sampling. All seed expansion uses the eight-stream XOF layout of Section 5.5. ExpandA expands ρ directly in the NTT domain: eight chunk streams apply G to ρ and a two-byte little-endian index, reject masked 16-bit words at least q, and fill the flattened matrix; scalar code stores canonical NTT order while vector code imports the same residues. SampleS draws $[f_0, \mathbf{s}_0, \mathbf{e}]$ from deterministic streams that feed RCDT words, sign bits, and a zero-fold bit to BaseSampler, and SampleY expands the mask from ρ'' and counter t, calling SampleZ with $k = 128$ per coefficient. The byte-exact pseudocode for all three is in the released reference code.

Canonical public vector and key encoding. Key generation computes $\mathbf{b}$ from $(\mathbf{A}_0, \mathbf{s}_0, \mathbf{e}, f)$, converts it through the scalar reference NTT, and serializes it in canonical order; optimized backends may permute it only after decoding, so backend-specific NTT layouts never become wire formats. Writing $\mathsf{Enc}_q, \mathsf{Dec}_q$ for modulo-q packing and $\mathsf{Enc}_\sigma, \mathsf{Dec}_\sigma$ for signed narrow-secret packing, the public key is ρ followed by $\mathsf{Enc}_q(\widehat{\mathbf{b}}[i])$ for $i < m$, and the secret key is $\rho \| tr \| K$ followed by Enc_σ of $f_0 = (f-1)/2$, of each $\mathbf{s}_0[j]$ and $\mathbf{e}[i]$, and of the cached $\widehat{\mathbf{b}}$; the decoders invert these field by field. The full pseudocode is in the released reference code.

rANS-backed signature packing. Signatures contain the fixed-length challenge $\tilde{c}$ followed by one deterministic two-state rANS stream for $\mathbf{h}$ and the high part of $\mathbf{z}_1$, written into the fixed reserve R with a two-byte length and zero padding, and then the verbatim low bits of $\mathbf{z}_1$ (Appendix B.2.4 gives the layout and size checks). The parser reverses this, rejecting any nonzero padding, an over-long length, or a malformed rANS stream. As with the key encodings, the byte-exact encoder and parser are in the released reference code.

Bucketing primitives. With $W_1 = 2(q-1)/\tau$, the rounding map is $\mathsf{HighBits}(w) = \lfloor ((w \bmod 2q) + \tau/2)/\tau \rfloor \bmod W_1$ and $\mathsf{LSB}(a) = (a \bmod 2q) \bmod 2$; MakeHint and UseHint below carry the one-bucket correction between signer and verifier.

Scalar IRS decision rule. This derives the one-dimensional form of the IRS decision rule that Algorithm 6 (Section 5.2) evaluates, starting from the vector-valued rule of Equations (4) and (5). For one step, let

$$u = \langle \mathbf{z}, \mathbf{v} \rangle / r^2, \qquad \gamma = \|\mathbf{v}\|^2 / (2r^2).$$

Then each ratio in the series $S_{\mathbf{v}}(\mathbf{z})$ of Equation (3) becomes

$$\frac{\rho_r(\mathbf{z} + k\mathbf{v})}{\rho_r(\mathbf{z})} = \exp(-ku - k^2\gamma),$$

so the series can be written as $S_\gamma(u) = \sum_{k \geq 0} (-1)^k e^{-ku - k^2\gamma}$,

Algorithm 7: MakeHint$(\mathbf{w}, \mathbf{z}_2)$

Input: commitment $\mathbf{w} \in R^m$, hidden response tail $\mathbf{z}_2 \in R^m$
Output: hint vector $\mathbf{h} \in R^m$

$W_1 \leftarrow 2(q-1)/\tau$
foreach *component i and coefficient k* **do**
  $\widetilde{w} \leftarrow \mathbf{w}[i][k] - 2\mathbf{z}_2[i][k] \bmod 2q$
  $\mathbf{h}[i][k] \leftarrow$ $(\mathsf{HighBits}(\mathbf{w}[i][k]) - \mathsf{HighBits}(\widetilde{w})) \bmod^{\pm} W_1$
return h

Algorithm 8: UseHint$(\widetilde{\mathbf{w}}, \mathbf{h}, w_0)$

Input: recomputed commitment $\widetilde{\mathbf{w}}$, hint $\mathbf{h}$, low bit vector w_0
Output: recovered high bits $\mathbf{w}_h$ and reconstructed tail $\mathbf{z}_2$

$W_1 \leftarrow 2(q-1)/\tau$
foreach *component i and coefficient k* **do**
  $\mathbf{w}_h[i][k] \leftarrow (\mathsf{HighBits}(\widetilde{\mathbf{w}}[i][k]) + \mathbf{h}[i][k]) \bmod W_1$
  $\delta \leftarrow [i = 0] w_0[k]$
  $\mathbf{z}_2[i][k] \leftarrow (\tau \mathbf{w}_h[i][k] - \widetilde{\mathbf{w}}[i][k] + \delta)/2 \bmod^{\pm} q$
return $(\mathbf{w}_h, \mathbf{z}_2)$

Algorithm 9: SampleC$(\tilde{c})$

Input: challenge seed $\tilde{c}$
Output: sparse challenge $c \in R$ with weight κ

$X \leftarrow \mathsf{H.Init}(); \mathsf{H.Absorb}(X, \tilde{c})$
$c \leftarrow 0 \in R$
for $i \leftarrow n - \kappa$ **to** $n - 1$ **do**
  repeat
    obtain a byte b from $\mathsf{H.Squeeze}(X)$
  until $b \le i$
  $c_i \leftarrow c_b$; $c_b \leftarrow 1$
return c

and Equations (4) and (5) become the one-dimensional rule

$$f(u) = \begin{cases} S_\gamma(u)/M, & u \ge 2\gamma, \\ (1 - S_\gamma(-u))/M, & u < 2\gamma, \end{cases}$$

$$g(u) = \begin{cases} (1 - S_\gamma(u))/M, & u \ge -2\gamma, \\ S_\gamma(-u)/M, & u < -2\gamma. \end{cases}$$

Section 5.2 evaluates S_γ by a Horner chain over the precomputed coefficients $c_k = (-1)^k e^{-k^2\gamma}$.

B Failure-Bound Derivations

B.1 Proofs of Approximation Theorems

Proof of Theorem 1. The BaseSampler step is the Rényi-divergence case of [30, Theorem 7] under (6), losing one bit. For ApproxExp, a rejection sampler emits z with probability proportional to $q(z)\,e^{-a(z)}$, where $q(z)$ is the ApproxExp-independent proposal mass and $e^{-a(z)}$ the ideal acceptance probability [30, Lemma 8]. The implementation uses $\widehat{e}(a(z)) = e^{-a(z)}(1+\theta_z)$, $|\theta_z| \le \eta$, so the emitted distribution is reweighted by $1+\theta_z$ and renormalized by the P-average $1+\bar{\theta}$ of the same factors, giving per-output relative error $|(1+\theta_z)/(1+\bar{\theta}) - 1| \le 2\eta/(1-\eta)$. Rejected candidates are resampled and never leave the constant-time sampler, so this is the only ApproxExp-dependent object the attack observes. By (7) and $Q_Z \le Q_{\exp}$, $\frac{2\eta}{1-\eta} \le 1/\sqrt{2(2\lambda-1)Q_{\exp}} \le 1/\sqrt{2(2\lambda-1)Q_Z}$, so probability preservation [30, Lemma 11] over the Q_Z emitted samples loses one further bit, for $(\lambda-2)$-bit security. □

Proof of Theorem 2. Let

$$S(w) = \sum_{k=0}^{\infty} (-1)^k e^{-kw-k^2\gamma},$$

$$\widehat{S}_T(w) = \sum_{k=0}^{T-1} (-1)^k e^{-kw-k^2\gamma}.$$

For one IRS step on input $\mathbf{y}$, write $u = \langle \mathbf{y}, \mathbf{v} \rangle / r^2$ and $\gamma = \|\mathbf{v}\|^2/(2r^2)$. The ideal kernel P is over the three outcomes $\{\mathbf{y}-\mathbf{v}, \mathbf{y}+\mathbf{v}, \perp\}$ with the probabilities $f, g, 1-f-g$ of Equations (4) and (5). The forced kernels $P^\star$, $P_T^\star$, and $\widehat{P}^\star$ are over the two outputs $\{\mathbf{y}-\mathbf{v}, \mathbf{y}+\mathbf{v}\}$: each keeps the $\mathbf{y}-\mathbf{v}$ probability f of its respective series evaluation (exact, truncated, or truncated fixed-point) and assigns all remaining mass to $\mathbf{y}+\mathbf{v}$.

Lemma 1 (Tail of the alternating IRS series)**.** *For every $w \ge -2\gamma$ and integer $T \ge 1$,*

$$|\widehat{S}_T(w) - S(w)| \le E_T(w) := e^{-Tw-T^2\gamma}.$$

Proof. The truncation error is

$$S(w) - \widehat{S}_T(w) = \sum_{k=T}^{\infty} (-1)^k a_k, \qquad a_k = e^{-kw-k^2\gamma} > 0.$$

The consecutive ratio is $a_{k+1}/a_k = e^{-(w+(2k+1)\gamma)}$, and $w \ge -2\gamma$, $k \ge T$ give $w + (2k+1)\gamma \ge (2T-1)\gamma > 0$, so the positive a_k decrease strictly from the first omitted term, and the alternating-series bound gives $|S(w) - \widehat{S}_T(w)| \le a_T = E_T(w)$. □

Lemma 2 (Per-output relative error of the forced truncation)**.** *For every input $\mathbf{y}$,*

$$\max_{o \in \{\mathbf{y}-\mathbf{v}, \mathbf{y}+\mathbf{v}\}} \left| \frac{P_T^\star(o)}{P^\star(o)} - 1 \right| \le \delta_{\mathrm{rel}} := \frac{e^{-(T^2-2T-3)\gamma}}{1-e^{-\gamma}}.$$

Proof. Both forced kernels output $\mathbf{y}-\mathbf{v}$ with the probability f computed from their series and $\mathbf{y}+\mathbf{v}$ with the complement, so it suffices to bound $|\Delta f|/f$ and $|\Delta f|/(1-f)$, where Δf is the change of f under truncation. By Lemma 1, every series evaluation used by the IRS rule has absolute error at most $E_T(w) = e^{-Tw-T^2\gamma}$ at its argument $w \geq -2\gamma$. Since the ideal step is a probability kernel by the choice of the repetition rate M [27], $f+g \leq 1$, so $1-f \geq g$ and we can reuse the standard lower bounds on the ideal probabilities f and g in each region of the IRS rule.

If $u \geq 2\gamma$, then $f = S(u)/M$ and $|\Delta f| \leq E_T(u)/M$, and the alternating-series bounds $S(u) \geq 1-e^{-3\gamma}$ and $1-S(u) \geq e^{-u-\gamma}(1-e^{-5\gamma})$ give $|\Delta f|/f \leq e^{-(T^2+2T)\gamma}/(1-e^{-3\gamma})$ and $|\Delta f|/(1-f) \leq E_T(u)e^{u+\gamma}/(1-e^{-5\gamma}) \leq e^{-(T^2+2T-3)\gamma}/(1-e^{-5\gamma})$, using $1-f \geq g = (1-S(u))/M$ and that $E_T(u)e^u$ decreases in u. The case $u < -2\gamma$ is symmetric, with the roles of the two outputs exchanged.

For $-2\gamma \leq u < 2\gamma$, $f = (1-S(-u))/M$ and $|\Delta f| \leq E_T(-u)/M \leq E_\star/M$ with $E_\star := E_T(-2\gamma) = e^{-(T^2-2T)\gamma}$. On $[-2\gamma, 2\gamma]$ the terms of $1-S(w)$ decrease ($w+3\gamma \geq \gamma > 0$), so $1-S(w) \geq e^{-3\gamma}(1-e^{-\gamma})$; hence $f, 1-f \geq e^{-3\gamma}(1-e^{-\gamma})/M$ and $|\Delta f|/f, |\Delta f|/(1-f) \leq E_\star e^{3\gamma}/(1-e^{-\gamma}) = e^{-(T^2-2T-3)\gamma}/(1-e^{-\gamma})$. Since $1-e^{-3\gamma}, 1-e^{-5\gamma} \geq 1-e^{-\gamma}$, this middle-region bound dominates both outer regions. Note no denominator of order $M-1$ appears: in the negligible-rejection regime $P(\perp) \approx (M-1)/M$, so a three-output relative gate including $\perp$ would be unprovable at our depths, and the forced kernel is the object matching both the implementation and the achievable bound. □

Lemma 3 (Pointwise relative error implies Rényi divergence)**.** *Let $\widehat{P}$ and P have the same support and satisfy $|\widehat{P}(o)/P(o) - 1| \leq \delta$ for all outcomes o. For any order $a > 1$,*

$$R_a(\widehat{P}, P) \leq \left(1 + \frac{a(a-1)\delta^2}{2(1-\delta)^{a+1}}\right)^{1/(a-1)}.$$

In particular, for small δ, $R_a(\widehat{P}, P) \leq 1 + a\delta^2 + O(a^2\delta^3)$.

Proof. This is Prest's second-order Rényi bound [42, Lemma 3]: writing $\widehat{P}(o) = P(o)(1+\theta_o)$ with $\sum_o P(o)\theta_o = 0$, the Taylor bound $(1+x)^a \leq 1 + ax + a(a-1)x^2/(2(1-\delta)^{a+1})$ summed with weights $P(o)$ cancels the linear term, and the $(a-1)$st root gives the claim. □

Throughout, M is Gärtner's repetition rate $M_\alpha = 1 + 2\alpha\sqrt{2\pi}\rho(\pi\alpha)/(\rho_\alpha(1)(1-\rho(2\pi\alpha)))$, $\alpha = r/\|\mathbf{v}\|$ (Corollary 1, i.e. Theorem 1 with the Lemma 3 bound) [27]; this is the smallest M for which the ideal step is a valid kernel, with $f, g \geq 0$ and $f+g \leq 1$, so $P(\perp) = 1-f-g \geq 0$ pointwise. The next lemma bounds that restart mass.

Lemma 4 (Pointwise restart mass)**.** *For every input $\mathbf{y}$, the ideal IRS step satisfies*

$$P(\perp) \leq \frac{M-1}{M} + \frac{\Theta_{\max}(\gamma)}{M},$$

where $\Theta_{\max}(\gamma) := 4\sqrt{\pi/\gamma}\, e^{\gamma - \pi^2/(4\gamma)}$. In particular $P(\perp) \geq 0$, consistently with $M = M_\alpha \geq 1 + \max_u(-\Theta(u))$.

Proof. If $u \geq 2\gamma$, then $f+g = \big(S(u)+1-S(u)\big)/M = 1/M$, so $P(\perp) = (M-1)/M$ exactly; the case $u < -2\gamma$ is identical with $S(-u)$ in place of $S(u)$. In the middle region, $P(\perp) = 1-(2-S(u)-S(-u))/M = \big((M-1)+\Theta(u)\big)/M$ with

$$\Theta(u) := S(u) + S(-u) - 1 = \sum_{k\in\mathbb{Z}} (-1)^k e^{-k^2\gamma + ku}.$$

Poisson summation applied to $x \mapsto e^{-\gamma x^2 + (u+i\pi)x}$ gives

$$\Theta(u) = \sqrt{\frac{\pi}{\gamma}}\, e^{u^2/(4\gamma)} \sum_{j\in\mathbb{Z}} e^{-\pi^2(2j-1)^2/(4\gamma)} \cos\frac{\pi(2j-1)u}{2\gamma}.$$

Bounding the cosines by 1 and the sum over j by $4e^{-\pi^2/(4\gamma)}$, and using $e^{u^2/(4\gamma)} \leq e^{\gamma}$ on $|u| < 2\gamma$, yields $\Theta(u) \leq \Theta_{\max}(\gamma)$. □

Lemma 5 (One-step divergence of the forced implementation)**.** *Let $a = 2\lambda - 1$ and consider an input with $|u| \leq u_{\max}$ for some $u_{\max} \geq 2 + 2\gamma$. Assume the truncation gate (10) with $\delta := 1/\sqrt{2(2\lambda-1)Q_{\text{step}}}$ and $a\delta \leq 1/4$, and the forcing gate (9) with $P(\perp) \leq \bar{\mu}$. Then*

$$R_a\big(\widehat{P}^\star, P\big)^{a-1} \leq 1 + \frac{a(a-1)\delta^2}{2(1-\delta)^{a+1}} + 3(a-1)\bar{\mu}.$$

Proof. Write $\mu := P(\perp)$ and $x := \mu/g$. On the window $g \geq e^{-u_{\max}-\gamma}(1-e^{-5\gamma})/M$ (the $u \geq 2\gamma$ bound of Lemma 2; the other regions are larger when $u_{\max} \geq 2+2\gamma$), so the second inequality of (9) gives $(a-1)x \leq (a-1)\bar{\mu}Me^{u_{\max}+\gamma}/(1-e^{-5\gamma}) \leq 1/2$. By the gate $\widehat{P}^\star(o) = P^\star(o)(1+\tau_o)$, $|\tau_o| \leq \delta$, with $f\tau_- + (1-f)\tau_+ = 0$; using $P^\star(\mathbf{y}-\mathbf{v}) = f = P(\mathbf{y}-\mathbf{v})$ and $P^\star(\mathbf{y}+\mathbf{v}) = 1-f = g(1+x)$, split $R_a^{a-1} = [f(1+\tau_-)^a + (1-f)(1+\tau_+)^a] + (1-f)(1+\tau_+)^a[(1+x)^{a-1}-1]$. The first bracket is $\leq 1 + a(a-1)\delta^2/(2(1-\delta)^{a+1})$ by Lemma 3; for the second, $(1+x)^{a-1} - 1 \leq \frac{3}{2}(a-1)x$, $(1-f)x \leq \frac{3}{2}\mu$, and $(1+\tau_+)^a \leq \frac{4}{3}$ multiply to $\leq 3(a-1)\bar{\mu}$. □

Proof of Theorem 2. The explicit truncation-only bound is Lemma 2. For the security statement, we compare the implemented scheme with the ideal scheme through window-capped versions of both: the capped variant of a scheme declares the signing attempt failed as soon as some IRS step receives an input with $|u| > u_{\max}$, and is unchanged otherwise. The capped kernels of the two schemes therefore agree outside the window and satisfy the hypotheses of Lemma 5 inside it.

We first bound the probability that the cap fires, using the standard left-to-right hybrid order (replacing the ideal IRS step before the ideal sampler). By the output-preservation property [27, Theorem 1], the accumulator stays distributed as $\mathcal{D}_{\mathbb{Z}^N, r}$ (N the mask dimension) at the start of every step of a

live attempt, so $\langle \mathbf{y}, \mathbf{v} \rangle$ is subgaussian with parameter $r\|\mathbf{v}\|$ [34] and $\Pr[|u| > u_{\max}] \le 2e^{-u_{\max}^2/(4\gamma)}$ per step; conditioning on a live attempt inflates this by $\le 1/(1-2^{-32})$ via (9). A union bound over $\le Q_{\text{step}}$ steps and the tail gate (8) bound the lifetime cap probability by $p_{\text{cap}} \le 2^{-\lambda-6}/(1-2^{-32}) \le 2^{-\lambda-5.9}$.

Next, the divergence. With $a = 2\lambda - 1$ and $a\delta = \sqrt{a/(2Q_{\text{step}})} \le 2^{-6.5}$ (from $Q_{\text{step}} \ge 2^{12}(2\lambda-1)$), so that $(1-\delta)^{-(a+1)} \le 1 + 2^{-6.3}$, Lemma 5 bounds the order-a divergence of each capped step by $(1 + \frac{a-1}{4Q_{\text{step}}}(1+2^{-6.3}) + 3(a-1)\bar{\mu})^{1/(a-1)}$, using $a(a-1)\delta^2/2 = (a-1)/(4Q_{\text{step}})$. Multiplying over $\le Q_{\text{step}}$ steps with $Q_{\text{step}}\bar{\mu} \le 2^{-32}$ bounds the capped implemented-versus-ideal divergence by $\bar{R} := \exp(\frac{1}{4}(1+2^{-6.3}) + 3\cdot 2^{-32}) \le 2^{0.37}$, following the probability-preservation argument of [30, Lemma 11]. Each scheme coincides with its capped variant until the cap fires, and λ-bit security gives ideal success $\varepsilon_{\text{id}} \le 2^{-\lambda}$. Applying the same probability-preservation step (exponent $1 - 1/a$, $\lambda \ge 64$) to the cap event and to the success event bounds the implemented-side cap probability by $(\bar{R}\, p_{\text{cap}})^{1-1/a} \le 2^{-\lambda-4.9}$ and the implemented success by $(\bar{R}(\varepsilon_{\text{id}} + p_{\text{cap}}))^{1-1/a} \le 2^{-\lambda+0.9}$, summing to $2^{-\lambda}(2^{0.9} + 2^{-4.9}) < 2^{-(\lambda-1)}$. Hence the implemented scheme is $(\lambda-1)$-bit secure. □

B.2 Component Instantiations

B.2.1 RCDT Tables for BaseSampler

The three levels use $\lambda = 125, 189, 268$, hence Rényi orders $a = 2\lambda - 1 = 249, 377, 535$; all BaseSampler tables are 93-bit reverse-CDT, truncated near ten standard deviations, generated deterministically and not reproduced here. The SampleS noise sampler uses $r_s = \sigma = 1$, with $R_{535}(\text{BaseSampler}(1,U), \mathcal{D}_{\mathbb{Z}^+,1}) \le 1 + 2^{-88.09}$. For SampleZ, $k = 128$ and $r = 182,204,257$ give $r_s = 1.421875, 1.59375, 2.0078125$ and $R_{249} \le 1 + 2^{-81.47}$, $R_{377} \le 1 + 2^{-83.39}$, $R_{535} \le 1 + 2^{-80.50}$.

B.2.2 ApproxExp Error Budget

The sampler ApproxExp returns a 63-bit fixed-point $\widehat{e}(a)$ for e^{-a}, $a = y(y+256x)/(2r^2)$ with $0 \le y < 128$. We validate it *end to end*—the released C-routine mirror on every reachable input, not the polynomial alone—against high-precision references.

Instantiation. The end-to-end scan gives the accept-branch error $\eta = \max_a |\widehat{e}(a)/e^{-a} - 1| \le 2^{-49.43}$ at all three levels, so the emitted-sample distortion of (7) is $2\eta/(1-\eta) \le 2^{-48.4}$, over five bits below the $2^{-43.3}$ gate; we use the conservative 2^{-48} in the main text.

The reject branch. The reject-branch ratio $|\varepsilon_{\text{exp}}/(1-e^{-a})|$ that the stronger Condition (1) of [30, Theorem 7] would also bound reaches $2^{-40.5}$ at the smallest queried a (where $1 - e^{-a} \approx a < 2^{-15}$), but this is harmless: Theorem 1 controls only the emitted distribution, never the internal accept/reject bit, which is neither part of the signature nor a function of the long-term secret. A higher-degree polynomial would meet Condition (1) too; we keep degree 6 because the emitted bound already suffices and preserves the default KATs.

Table 6: Average CPU cycles per exponential evaluation.

Method	Cycles
Falcon 12-degree polynomial	60.0
Our 4x method	35.1
AVX2 method	6.8
AVX-512IFMA method	5.7

† Each benchmark runs more than 1000 times; we report median cycles.

Per-call cycle counts. Table 6 gives the median per-call cost (Section 5.1.2) behind the 1.7×, 8.8×, and 10.5× speedups over Falcon's degree-12 polynomial cited in Section 5.

B.2.3 Instantiating the IRS Truncation Gates

LITHIUM uses the negligible-rejection regime of the IRS parameters, and we instantiate the three gates of Theorem 2 with $u_{\max} = 8$ and the lifetime count $Q_{\text{step}} = 2^{69.9}$ from Table 3. The side conditions hold with large margins: $\lambda \ge 125 \ge 64$, $Q_{\text{step}} \ge 2^{12}(2\lambda-1)$, and $u_{\max} = 8 \ge 2 + 2\gamma$.

Forcing and tail gates. The ideal repetition rates $M - 1 = 2^{-107.0}, 2^{-108.2}, 2^{-107.8}$ keep the per-signature restart probability below 2^{-101}, so the implementation omits the restart branch; with $\Theta_{\max} \le 2^{-106.0}, 2^{-107.2}, 2^{-106.7}$ from Lemma 4 at the upper norm endpoint, $\bar{\mu} \le 2^{-105}$, and both inequalities of (9) hold with vast margin. The per-step tail $2e^{-16/\gamma_{\max}} \le 2^{-721}$ gives lifetime mass $\le 2^{-651} \ll 2^{-\lambda-6}$ in (8).

Truncation gate and the choice of T. The lower norm endpoint $\gamma_{\min} = 0.02990, 0.02990, 0.03029$ gives the slowest convergence. Validating the fixed-point kernel against a real-arithmetic infinite-series reference over $|u| \le u_{\max}$ at both endpoints, $T = 33$ gives worst two-sided relative deviations $2^{-43.2}, 2^{-43.2}, 2^{-43.8}$ on the binding window $|u| \le 4\gamma$ (and $\le 2^{-48.8}$ deeper)—over three bits below the $2^{-40.0}$ gate—whereas $T = 32$ gives only $2^{-40.6}, 2^{-40.6}, 2^{-41.1}$, so $T = 33$ is the first adequate depth.

B.2.4 Signature Size and rANS Overflow

The signature is $\sigma = \tilde{c} \,\|\, \mathsf{LE}_{16}(L) \,\|\, S \,\|\, 0^{R-L} \,\|\, \mathsf{Enc}_{b_{\text{lo}}}(\mathbf{z}_{1,\ell})$: the n-bit challenge $\tilde{c}$, a merged two-state rANS stream S for the rounded heads of $\mathbf{z}_1$ and the hint $\mathbf{h}$ in a fixed reserve R, and the $b_{\text{lo}} = \lfloor \log_2 r \rfloor$ verbatim low bits per $\mathbf{z}_1$ coefficient, so

$$|\sigma| = \lceil n/8 \rceil + 2 + R + (\ell+1)\lceil n b_{\text{lo}}/8 \rceil. \tag{13}$$

Table 7: The LITHIUM-sUF reference parameter sets.

Parameter	LITHIUM-sUF-120	LITHIUM-sUF-180	LITHIUM-sUF-260
n, q	$256, 417793$	$256, 216577$	$512, 617473$
ℓ, m	$3, 2$	$4, 3$	$2, 2$
κ, τ	$30, 512$	$58, 1024$	$58, 512$
σ, r	$1, 193$	$1, 214$	$1.7, 436$
B'_k, B_k	$47.3, 48.7$	$52.4, 53.8$	$107.5, 110.1$
B_s, B_v	$9233, 11249$	$11527, 16322$	$25804, 28522$
NTWE/SIS-UF/sUF bits	$147/153/125$	$205/224/185$	$269/316/265$
pk/sig bytes	$1248/1528$	$1760/1909$	$2592/2817$
pk+sig bytes	2776	3669	5409

† Symbols are as in Table 2. Sizes use the same encoding and (13), with the rANS reserve sized so that $\Pr[\text{overflow}] \le 2^{-101}$; the resulting slack over the Shannon entropy floor is 142, 185, and 307 bytes for the three security levels. Unlike Table 2, every modulus here exceeds 2^{16}, so the 16-bit vectorized NTT path does not apply; at the 260 level no choice below 2^{16} meets all three constraints at once.

* Security estimates are classical lattice-estimator [3, 4] outputs for the NTWE, SIS-UF, and SIS-sUF bounds according to [27], where the rank-k NTWE is approximately estimated using the rank-$(k+1)$ Module-LWE. The binding target is sUF: since the sUF length bound $2B_v$ is the looser one, SIS-UF is implied and is reported only for comparison. The search additionally enforces $2B_v < q$, without which the homogeneous MSIS instance that the sUF reduction produces admits the trivial solution $q\mathbf{e}_i$; here $2B_v/q$ is 0.05, 0.15, and 0.09.

The rANS stream (precision 2^{10}) codes over the true emitted PMFs: the $\mathbf{z}_1$ head $Z_h = \lfloor (Z + 2^{b_{\text{lo}}-1})/2^{b_{\text{lo}}} \rfloor$, $Z \leftarrow \mathcal{D}_{\mathbb{Z},r}$, and the hint bucket-crossing variable from $\mathcal{D}_{\mathbb{Z},r}$. This is where our accounting differs from Gärtner's Appendix A.2 [27], which models MakeHint as Gaussian and can give negative entropy contributions in small-variance regimes. The gap between the Shannon entropy floor and (13) is rANS quantization, cross-entropy loss, the flush, byte rounding, and the zero-padded reserve, which a Bernstein bound sizes so that $\Pr[\text{overflow}] \le 2^{-101}$.

B.2.5 A Strongly Unforgeable Parameter Set

Table 2 targets SIS-UF, so its sUF estimates fall below the level. For readers who want the level to hold in the strong sense, Table 7 reruns the search of Section 3.1 with SIS-sUF as the binding target, keeping every other choice identical: the same ≥ 5-bit margin, the same $\sigma \ge 1$, the same restart gates, and the same encoding, so the two tables are directly comparable. We name the result LITHIUM-sUF-120/180/260 to keep it distinct from the LITHIUM instantiation of Table 2, which is what every other section of this paper means by LITHIUM.

Two things change. First, the sUF reduction subtracts two accepted signatures and lands on a *homogeneous* MSIS instance of norm at most $2B_v$, for which $q\mathbf{e}_i$ is a legal short solution; the estimate is therefore meaningful only when $2B_v < q$, which tightens the usual $B_v < q$ convention by a factor of two. Second, absorbing a doubled length bound is cheapest through the SIS rank mn, which moves the module shape away from that of Table 2. Together these cost 533, 579, and 981 bytes of pk + sig at the three levels, about 21% overall, and push the moduli past 2^{16}. We give LITHIUM-sUF as a reference point rather than as our headline instantiation; Section 6 continues to use Table 2.

B.2.6 Reference-Implementation Performance

Section 6 focuses on the optimized backends (Table 5); Table 8 gives the corresponding portable-C reference-implementation numbers on the same Intel Rocket Lake host, using the same StQ_2/tail/median methodology of Section 6.1. For reference implementations, LITHIUM signs faster than both ML-DSA and HAETAE, and its tail stays near one where ML-DSA's is 3–4× p50.

B.2.7 Cross-Platform Benchmark on AMD Zen 5

This expands the AMD Zen 5 cross-check of Section 6.2 with the complete benchmark tables, gathered on a Google Cloud C4D instance (AMD EPYC 9B45, Zen 5) with the same StQ_2/tail/median methodology of Section 6.1. Table 9 gives the optimized-backend numbers and Table 10 the reference-implementation numbers, mirroring Tables 5 and 8. In StQ_2, LITHIUM-120's fastest signer takes 95k cycles against 113k for ML-DSA-44 AVX2 and 813k for HAETAE-120 AVX2, and the KAT-locked AVX-512 default (117k) matches ML-DSA AVX2; across all three levels, LITHIUM's fastest signer is 1.1–1.2× faster than ML-DSA and 5.7–8.6× faster than HAETAE, and for reference implementations LITHIUM again signs faster than both ML-DSA and HAETAE with a near-one tail.

B.2.8 Secure-Implementation Audit Methodology

This expands the static-analysis methodology behind Section 6.4. We first separate secret from public data and mark every declassification point across the signing flow (the point after which a value—e.g., the accept/reject bit of a rejection sampler, or a hint—is no longer treated as sensitive). We then inspect the compiler-generated assembly for every combination of the three backends (reference, AVX2, AVX-512), the three security levels, two compilers (GCC and Clang),

Table 8: Reference-implementation performance.

Scheme	KeyGen	Sign	Tail	Verify
LITHIUM-120	890k	972k	1.02	171k
LITHIUM-180	1269k	1585k	1.02	272k
LITHIUM-260	1810k	2045k	1.01	307k
ML-DSA-44	287k	1047k	4.36	315k
ML-DSA-65	526k	1665k	4.14	504k
ML-DSA-87	769k	2017k	3.75	806k
HAETAE-120	1047k	3342k	6.15	212k
HAETAE-180	1682k	4312k	5.19	373k
HAETAE-260	1486k	6222k	5.87	461k
HAWK-512	9096k	353k	1.02	326k
HAWK-1024	44588k	761k	1.02	703k
NTRU+Sign-512	652k	1147k	4.60	71k
NTRU+Sign-1024	1328k	3462k	6.43	134k
Raccoon-128	3094k	5501k	1.02	1863k
Raccoon-192	4697k	7925k	1.02	2978k
Raccoon-256	7325k	11715k	1.01	5028k

* The suffix k denotes thousands of CPU cycles. KeyGen and Sign are StQ_2; Tail is p99/p50 for signing time; Verify is the median. Code: ML-DSA https://github.com/pq-crystals/dilithium@bba9534; HAETAE https://github.com/kpqc-cryptocraft/KpqClean_ver2@0896ebe; HAWK https://github.com/hawk-sign/dev@1b9fef5; NTRU+Sign https://github.com/KU-Cryptographic-Protocol-Lab/ntruplus_sign@4a540a; Raccoon https://github.com/masksign/raccoon@e789b4b.

Table 9: Optimized-implementation performance on AMD Zen 5.

Scheme	Impl.	KeyGen	Sign	Tail	Verify
LITHIUM-120	AVX2	148k	177k	1.04	37k
LITHIUM-180	AVX2	204k	286k	1.04	55k
LITHIUM-260	AVX2	305k	369k	1.04	67k
LITHIUM-120	AVX2Exp	146k	169k	1.14	37k
LITHIUM-180	AVX2Exp	204k	284k	1.05	55k
LITHIUM-260	AVX2Exp	310k	351k	1.04	68k
LITHIUM-120	AVX-512	105k	117k	1.09	36k
LITHIUM-180	AVX-512	142k	188k	1.06	50k
LITHIUM-260	AVX-512	221k	234k	1.05	64k
LITHIUM-120	IFMAExp	104k	95k	1.03	36k
LITHIUM-180	IFMAExp	143k	159k	1.07	50k
LITHIUM-260	IFMAExp	220k	204k	1.06	64k
ML-DSA-44	AVX2	47k	113k	4.10	52k
ML-DSA-65	AVX2	80k	181k	3.99	84k
ML-DSA-87	AVX2	129k	225k	3.33	141k
HAETAE-120	AVX2	258k	813k	6.10	60k
HAETAE-180	AVX2	359k	905k	5.17	85k
HAETAE-260	AVX2	392k	1221k	5.82	106k
HAWK-512	AVX2	2960k	49k	1.02	60k
HAWK-1024	AVX2	17211k	100k	1.03	124k
Falcon-512	SSE2	5864k	462k	1.04	25k
Falcon-1024	SSE2	21851k	926k	1.03	45k
Falcon-512	AVX2	–	388k	1.04	–
Falcon-1024	AVX2	–	807k	1.03	–
Falcon-512	AVX-512	–	366k	1.04	–
Falcon-1024	AVX-512	–	742k	1.03	–

* Columns, metrics, and code sources are as in Table 5; this table repeats the same benchmark on a Google Cloud C4D instance (AMD EPYC 9B45, Zen 5) instead of the Intel Rocket Lake host of Section 6.1.

two optimization levels ($-O3$ and $-Os$), and the default versus opt-in ApproxExp variants, checking each configuration for secret-dependent branches or memory addressing and for high-risk instructions: division, square root, secret-indexed gathers/scatters, and secret-indexed table lookups. Across this matrix, the only occurrences of these patterns are public SHAKE lane operations (indexed by public round/lane counters) or divisions whose divisor is a public length or constant, never signing-secret-dependent.

C Ethical Considerations

This work designs, proves, and benchmarks a lattice-based signature scheme; it does not collect human-subject data, measure live third-party systems or users, or disclose vulnerabilities in deployed software. Section 8 already summarizes this scope; we expand it here per the call's guidance to keep a well-labeled ethics discussion.

Hardness-assumption maturity. LITHIUM relies on NTWE, a combination of the NTRU and LWE problems [26] used by Gärtner's IRS construction [27]. Existing evidence relates rank-k NTWE to both rank-k Module-LWE and NTRU-style hardness: NTWE admits a q-ary lattice formulation whose primal and dual attack surfaces mirror well-studied LWE and NTRU variants, enabling concrete hardness estimates with the same BKZ/Core-SVP methodology used for deployed lattice schemes, and the rank-k NTWE primal lattice has essentially the same determinant and target-vector geometry as rank-$(k+1)$ Module-LWE. A reasonable concern is that mature cryptographic systems should ideally rest on mature and extensively studied assumptions; our response is that the main implementation techniques in this work are not specific to NTWE—the Gaussian-sampling, ApproxExp, IRS-decision,

Table 10: Reference-implementation performance on AMD Zen 5.

Scheme	KeyGen	Sign	Tail	Verify
LITHIUM-120	495k	533k	1.03	98k
LITHIUM-180	706k	880k	1.03	157k
LITHIUM-260	978k	1189k	1.02	164k
ML-DSA-44	155k	558k	4.40	169k
ML-DSA-65	297k	922k	4.18	270k
ML-DSA-87	414k	1132k	3.96	435k
HAETAE-120	536k	1757k	6.11	115k
HAETAE-180	873k	2280k	4.74	196k
HAETAE-260	736k	3219k	5.89	242k
HAWK-512	5210k	174k	1.07	185k
HAWK-1024	26156k	378k	1.04	395k
NTRU+Sign-512	416k	648k	4.59	41k
NTRU+Sign-1024	852k	1980k	6.44	75k
Raccoon-128	1803k	3355k	1.03	1130k
Raccoon-192	2769k	4832k	1.01	1842k
Raccoon-256	4336k	7173k	1.02	3088k

* Columns, metrics, and code sources are as in Table 8; this table repeats the same benchmark on a Google Cloud C4D instance (AMD EPYC 9B45, Zen 5) instead of the Intel Rocket Lake host of Section 6.1.

and vectorization optimizations apply to any scheme that uses iterative rejection sampling and Gaussian sampling, so if future work replaces NTWE by Module-LWE in an IRS-based construction, these performance techniques should remain applicable.

Research subjects and data. No human participants, personal data, or production systems are involved. All inputs to our experiments are synthetic (random seeds, messages, and keys generated for benchmarking) or drawn from the public reference implementations of ML-DSA, Falcon, and HAETAE that we compare against.

Dual-use and disclosure. LITHIUM is a constructive proposal, not an attack on a deployed system. The threat-model boundary in Section 1 is stated explicitly (timing/cache leakage in the signing core; physical side channels and faults are out of scope) so that readers do not over-read our secure-implementation evidence as a broader guarantee. We are not aware of any responsible-disclosure obligation triggered by this work.

Benchmarking practice. All comparisons in Section 6 use the reference or vectorized implementations that the respective schemes' authors publish, compiled and run under the same methodology on machines we control; we do not benchmark or characterize systems we do not own or have permission to measure.

Generative AI. As stated in Section 8, generative AI assisted editorial polishing and test-code drafting only; every such output was reviewed and validated by the human authors, who take full responsibility for the paper's technical content, code, and results.

D Open Science Appendix

Following the Open Science policy of USENIX Security '27, this appendix describes the artifacts that support the claims in this paper and how reviewers and readers can access them.

What is released. The portable reference implementation and the AVX2 and AVX-512 vectorized backends for all three LITHIUM security levels, covering key generation, signing, and verification; the deterministic known-answer tests (KATs) for the default and byte-exact backends, plus the separate KATs for the opt-in AVX2 and AVX-512IFMA ApproxExp variants (Section 5.1.2); the benchmark harness and raw logs behind Section 6 (Intel Rocket Lake and AMD Zen 5 platforms); and the secure-implementation evidence, i.e., the dudect and TIMECOP/KyberSlash-style harnesses and their raw traces referenced in Section 6.4.

Anonymous access. For review, the artifact is hosted at `https://github.com/Ji-Peng/Lithium-Code-Artifact`.

Public availability on acceptance. We intend to make the full artifact publicly available under a non-anonymous, stable link at camera-ready time, and to register it with the Artifact Evaluation Committee for functionality and reproducibility checking.