Evaluating Hybrid KEM/DSA for KpqC and NIST PQC on ARM Cortex-M4

Minjoo Sim[1][0000-0001-5242-214X], Minwoo Lee[1][0000-0002-2356-3055], Subeen Cho[2][0009-0004-4352-8021], Yulim Hyoung[2][0009-0009-7530-6497], and Hwajeong Seo[1][0000-0003-0069-9061]*

[1] Department of Information Computer Engineering, Hansung University, Seoul (02876), South Korea
{minjoos9797, minunejip}@gmail.com
[2] Department of Convergence Security, Hansung University, Seoul (02876), South Korea
{yulim4hyoung, chosubin1208}@gmail.com

Abstract. Primitive-only PQC benchmarks are insufficient for attributing composed hybrid costs on Cortex-M4 because shared hash backends, randomized-signature behavior, and fixed classical/wrapper work affect measured performance. We implement a common bare-metal Cortex-M4 harness for representative KpqC/NIST families, measuring uniform Hash-CT hybrid KEM benchmark rows with X25519 and Bindel et al. hybrid-signature AND-combiner rows. The goal is composed-cost attribution under a uniform benchmark transcript rather than primitive-only ranking.

Our measurements show three attribution effects. First, replacing only Keccak-$f[1600]$ changes SHAKE-heavy signing by up to 2.16×, while SPHINCS$^+$-SHA2 and FN-DSA control rows remain at 1.00×. Second, median-only signature tables can change deployment conclusions. In the nominal level-5 signing rows, HAETAE5 beats FN-DSA-1024 by median and mean latency, but its observed maximum reaches 4.53× its median while FN-DSA-1024 remains essentially flat. Third, a local SMAUG-T backend improves standalone SMAUG-T by 1.79–1.82×, but the visible gain drops to 1.42–1.65× inside SMAUG-T+X25519 hybrids. Supporting KEM rows place lattice hybrids at 3.7–8.8 M cycles and HQC hybrids at 19.6–74.5 M cycles. Together, the results motivate reporting composed-cost attribution, backend provenance, and variance alongside primitive timings.

Keywords: Post-Quantum Cryptography · Hybrid KEM · Hybrid DSA · ARM Cortex-M4 · Embedded Benchmarking · KpqC

* Corresponding author. hwajeong84@gmail.com

1 Introduction

Post-quantum deployment is now an integration problem, not only a primitive-selection problem. On microcontrollers, the combiner, transcript, byte layout, classical partner, and shared low-level backend can materially affect latency, wire size, stack, and flash [13,27]. This paper shows that Cortex-M4 hybrid post-quantum cryptography (PQC) cannot be understood from primitive timings alone, because shared symmetric backends, randomized-signature tails, and optimization dilution can change composed-cost conclusions even when the component primitives are fixed.

We evaluate this attribution layer for representative families selected by the National Institute of Standards and Technology (NIST) and the Korean Post-Quantum Cryptography (KpqC) competition. NIST has finalized ML-KEM, ML-DSA, and SLH-DSA [23,24,25], while FN-DSA and HQC have been selected for ongoing NIST standardization [28,26]. KpqC has selected SMAUG-T, NTRU+, HAETAE, and AIMer [17]. These families cover lattice, code-based, multi-party computation (MPC)-in-the-head, and hash-based designs, making them a useful portfolio for studying composed cost, backend sensitivity, and randomized-tail behavior.

The construction split is explicit. The key encapsulation mechanism (KEM) rows use a uniform ciphertext-hashing (Hash-CT) benchmark transcript with X25519, fixed encodings, labels, selected KpqC/NIST KEM components, and the Quantum Superiority Fighter (QSF) component interface. Section 2.1 defines the exact measurement-interface scope and X-Wing/QSF background. The DSA rows implement Bindel et al.'s hybrid signature AND-combiner with domain-separated component signatures [4]. Unlike prior embedded hybrid-PQC studies that mainly target Transport Layer Security (TLS)/X.509 stacks [6,14,31], this paper measures the primitive/harness layer and focuses on composed-cost attribution under one Cortex-M4 harness.

Contributions. This paper makes three contributions.

1. **Composed-cost attribution for hybrid PQC.** We implement a common Cortex-M4 harness for uniform Hash-CT benchmark rows and Bindel et al. hybrid signature AND-combiner rows, fixing byte layouts, labels, interfaces, backends, and target configuration so that classical-leg cost, wrapper work, and shared backend effects are visible.
2. **Backend sensitivity and randomized-tail measurements.** We show that a Keccak-$f[1600]$ backend swap changes SHAKE-heavy rows while control rows remain unchanged, and that rejection-sampling signatures require variance reporting beyond medians. We also report cycles, wire sizes, peak stack, code estimates, and selected sanity checks.
3. **SMAUG-T backend and optimization dilution.** We contribute a local Cortex-M4 SMAUG-T arithmetic backend and use it as a case study showing that standalone speedups shrink inside SMAUG-T+X25519 hybrids because fixed classical and wrapper work remain.

Section 2 defines the measured constructions and security scope. Section 3 describes the implementation and measurement setup. Section 4 presents the deployment results and backend case studies, and Section 5 concludes.

2 Preliminaries and Related Work

2.1 Hash-CT QSF Benchmark Rows

Barbosa et al. proposed X-Wing, an ML-KEM-768/X25519 KEM built from the QSF combiner [2]. The Internet Engineering Task Force (IETF) draft specifies ML-KEM-768/X25519 X-Wing [12], and the X-Wing KEM team publishes a C implementation [33]. The X-Wing paper also studies a Hash-CT transcript that includes the post-quantum ciphertext. We use Hash-CT QSF benchmark rows only as a local measurement label for the uniform transcript of Equation (1), not as a new KEM construction, a new combiner, or a measurement of the IETF X-Wing transcript. The measured Hash-CT benchmark rows hash the post-quantum ciphertext together with the component shared secrets and classical transcript fields. This keeps ML-KEM, SMAUG-T, NTRU+, and HQC under one benchmark interface, but the rows should not be read as implementing the IETF X-Wing wire transcript. In particular, the IETF ML-KEM-768/X25519 X-Wing transcript omits the ML-KEM ciphertext from the final combiner hash, whereas our Hash-CT rows include ct_{pq}. For large-ciphertext KEMs such as HQC, this adds an additional Hash-CT hashing cost relative to a no-ct_{pq} transcript. The DSA rows are separate and implement Bindel et al.'s hybrid signature AND-combiner as treated in Section 2.2.

For a selected algorithm pair, public keys, secret keys, and ciphertexts are fixed concatenations of the post-quantum and classical byte strings. The measured Hash-CT transcript computes the final shared secret as

$$ss = \mathsf{SHA3\text{-}256}(label \parallel ss_{\mathrm{pq}} \parallel ss_{\mathrm{cl}} \parallel ct_{\mathrm{pq}} \parallel ct_{\mathrm{cl}} \parallel pk_{\mathrm{cl}}), \tag{1}$$

where ss_{pq} and ss_{cl} are component shared secrets, ct_{pq} and ct_{cl} are component encapsulation tokens, and pk_{cl} is the long-term classical public key. The label has a fixed prefix and both component names, and fixed field lengths make parsing injective within each pair. The post-quantum public key is not hashed as a separate field, so adding it would define a different pk-bound benchmark point.

Security scope. The tables quantify benchmark costs for the Hash-CT benchmark rows above. The cited X-Wing/QSF work provides construction background for the component flow and interface. This paper does not derive fresh reductions for every substituted KpqC/NIST KEM component and does not claim side-channel, fault, or formal constant-time security. Including ct_{pq} selects the Hash-CT transcript measured here.

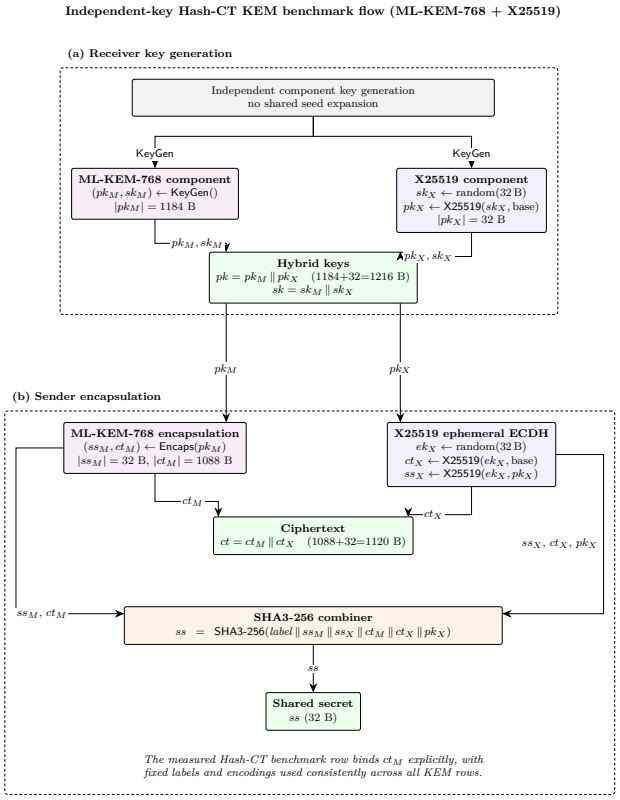


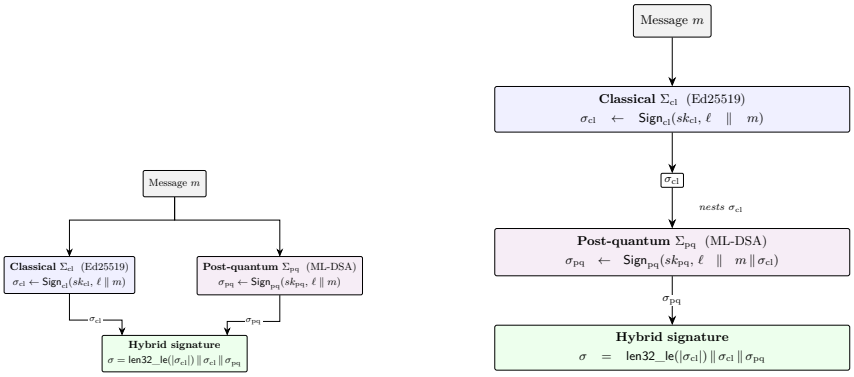
Fig. 1: X-Wing/QSF component flow adapted for the Hash-CT KEM benchmark rows measured in this paper. The ML-KEM-768/X25519 pair is shown as the baseline component pair. KpqC/NIST KEM rows use the same wrapper interface, fixed encodings, X25519 leg, and SHA3-256 Hash-CT transcript.

2.2 Hybrid Signature Combiner

For signatures, we implement the AND-combiner of Bindel et al. [4], where the classical and post-quantum schemes sign domain-separated inputs and verification accepts only if both component signatures verify. Let Σ_1 be the classical scheme and Σ_2 the post-quantum scheme. With fixed label ℓ, the primary concat mode is

$$\sigma_1 \leftarrow \mathsf{Sign}_1(sk_1, \ell \,\|\, m), \qquad \sigma_2 \leftarrow \mathsf{Sign}_2(sk_2, \ell \,\|\, m). \tag{2}$$

The hybrid signature is serialized with a 32-bit little-endian length prefix, denoted LE_{32}, as $\sigma = \mathsf{LE}_{32}(|\sigma_1|) \,\|\, \sigma_1 \,\|\, \sigma_2$. The length prefix makes parsing unambiguous. A nested mode is implemented only for overhead checks. In that mode the post-quantum input is $\ell \,\|\, m \,\|\, \sigma_1$. If concat and nested modes are used in one protocol context, labels must be mode-specific.



(a) Concat mode, where both components sign $\ell \parallel m$.

(b) Nested mode, where the post-quantum input also contains σ_{cl}.

Fig. 2: Bindel et al. hybrid signature combiners as implemented in this paper. Concat is the primary mode, and nested rows check input-processing overhead.

2.3 Related Work

Related work falls into three groups. X-Wing, Starfighters, RFC 9794, NIST transition guidance, and Bindel et al. provide the hybrid-construction background [2,12,11,13,27,4]. Protocol-level studies measure Transport Layer Security (TLS)/X.509 PQC or hybrid costs, including RLWE-TLS, Kyber/SPHINCS+ in mbed TLS, KEMTLS versus PQ-TLS, and KpqC in TLS/OpenSSL [5,6,14,31]. The closest KpqC-in-TLS line targets Cortex-A76 with OpenSSL, whereas we measure the bare-metal Cortex-M4 primitive/harness layer. Finally, pqm4, PQClean, and scheme-specific M4 work supply the backend lineage [15,30,7,18,10,22]. We fix the KEM wrapper, classical leg, transcript layout, footprint accounting, and shared backend before adding any TLS stack. This makes backend sensitivity, randomized-tail behavior, and optimization dilution visible below the protocol layer.

3 Implementation and Experimental Setup

3.1 Implementation Design

The harness separates Hash-CT KEM orchestration from Bindel et al. hybrid signature orchestration. KEM wrappers call component key generation, encapsulation, and decapsulation, concatenate post-quantum (PQ) bytes before classical bytes, and apply the SHA3-256 Hash-CT transcript of Equation (1). Signature wrappers implement the Bindel et al. concat AND-combiner as in Equation (2), and nested mode is retained only for overhead checks. Hybrid signatures are serialized as $\mathsf{LE}_{32}(|\sigma_{cl}|) \parallel \sigma_{cl} \parallel \sigma_{pq}$, with sizes taken from the selected algorithm descriptors.

Post-quantum components come from KpqC/NIST source trees, pqm4 [15], PQClean [30], and public Cortex-M4 repositories. We use optimized Cortex-M4 backends where available and identify portable or reference code when it

remains on the measured path. Table 1 lists the measured backend choices and provenance references. The pqm4 hardware abstraction layer (HAL) is the target abstraction used by the harness.

Table 1: Measured component backends and source references under the common Cortex-M4 harness.

Component	Measured backend	References
KEM/DSA orchestration	Hash-CT KEM wrapper and Bindel et al. DSA AND-combiner wrapper/dispatch	[2,33,4]
Harness/HAL	pqm4 HAL + SysTick harness	[15]
ML-KEM	PQClean/pqm4 `m4fspeed/m4fstack`	[23,30,15]
ML-DSA	PQClean/pqm4 `m4f`	[24,30,15]
FN-DSA	mupq/pqm4 Falcon/FN-DSA line	[28,29]
SPHINCS$^+$	PQClean portable scheme code + ARMv7-M SHA-256 or pqm4 Keccak	[25,30]
HQC	v5-compatible additive fast Fourier transform (FFT) M4 backend	[26,7]
SMAUG-T	KpqClean Ver2 + local `m4opt` polymul assembly	[16,9]
NTRU+	KpqClean Ver2 + public Cortex-M4 number-theoretic transform (NTT) backend	[16,10]
HAETAE	KpqClean Ver2 + Cortex-M4 backend	[16,8]
AIMer	KpqClean Ver2 port of streaming M4 backend	[16,18]
Classical curves	Lenngren X25519/P-256 assembly, TweetNaCl Ed25519 C, and mbedTLS-C ECDSA-P256	[20,19,21,3]
Symmetric	assembly Keccak-f[1600] + ARMv7-M SHA-256	[1,32]

NIST and KpqC components are integrated under the same bare-metal harness, toolchain, timing method, and backend-selection rule. Standalone probes check that intended backend paths are linked.

Artifact provenance and availability. KEM wrapper provenance starts from the public X-Wing C implementation at commit `c696a197` [33], adapted to pqm4, KpqC/NIST dispatch, and the measured Hash-CT benchmark transcript. For pqm4/mupq/PQClean-derived paths, the provenance audit records commit identifiers pqm4 `a24bb4b6`, mupq `ddccced`, and PQClean `c3e6861`. For KpqClean Ver2 components, HQC, classical curves, local wrappers, and plotting code, the artifact records backend identifiers or source snapshots, build flags, generated comma-separated value (CSV) files and log files, stack-watermark outputs, object-size summaries, and scripts used to regenerate the reported tables and figures.

We contribute local arithmetic only for SMAUG-T. The backend replaces the reference C polynomial-multiplication path for the NTT-unfriendly SMAUG-T ring [9] with hand-written ARMv7-M Toom-Cook/Karatsuba code, and deterministic test vectors match the reference backend. The kernel avoids intentional secret-dependent branches or table addressing, but no formal leakage analysis is claimed. Since prior Cortex-M4 SMAUG-T work exists [22], we use this backend to study optimization dilution inside a hybrid KEM stack.

The library uses static allocation where possible. Temporary wrapper buffers have compile-time maximum sizes. Scheme-owned code size is reported as an object-level estimate that sums the selected PQ `.text` sections and the measured classical-leg and combiner objects, while excluding shared Keccak, HAL, and harness code. These estimates are therefore incremental code-attribution numbers, not complete linked firmware images.

3.2 Target, Toolchain, and Reproducibility

The target is bare-metal ARM Cortex-M4F through the pqm4 HAL. Primary tables use arm-none-eabi-gcc 13.2.1, assembly (ASM) Keccak-$f[1600]$ unless stated otherwise, and mbedTLS 3.6.3 where no dedicated Cortex-M4 classical backend is available. The primary campaign uses the NUCLEO-L4R5ZI target at 20 MHz/0 flash wait-states with release build flags `-mcpu=cortex-m4 -mthumb -mfloat-abi=hard -mfpu=fpv4-sp-d16 -O3 -fomit-frame-pointer`. Cycle counts are the primary comparison metric. Wall-clock values are reported only for intuition and scale linearly with frequency only when wait-state and clock-dependent effects remain fixed. Timing uses pqm4 SysTick with overflow tracking and 64-bit cycle values. Functional validation checks component known-answer tests (KATs), hybrid shared-secret agreement, and SMAUG-T reference/ASM test-vector agreement.

3.3 Keccak Implementation Variants

The Keccak experiment compares the portable C Keccak-$f[1600]$ permutation with the pqm4 ARMv7-M backend [1]. Only the innermost permutation changes. Absorb, squeeze, padding, arithmetic kernels, classical code, target configuration, and harness stay fixed. Ratios therefore attribute cost to the shared Keccak permutation within this stack, rather than to changes in higher-level SHAKE/SHA3 code or arithmetic kernels.

3.4 Measurement Methodology

Timings are pqm4 HAL central processing unit (CPU) cycle counts. The HAL extends the 24-bit SysTick counter with overflow tracking and returns 64-bit values, and sample arrays use the same width. Primary KEM rows report $N = 100$ medians. Lattice/FN-DSA rows with randomized or rejection-sampling behavior also report $N = 100$ medians, and Table 5 adds mean, minimum, maximum, and max/min spread summaries from the retained measurement logs. These summaries are intended to expose median-hidden tail behavior, but they are not 90th/95th-percentile (p90/p95) estimates and are not worst-case execution-time bounds.

AIMer and SPHINCS$^+$ use smaller repetition counts, as stated in the corresponding table notes. Those rows support coarse deployment and backend-attribution comparisons. Distributional claims are limited to the $N = 100$ rows.

4 Deployment Evaluation

This section reports NUCLEO-L4R5ZI measurements at 20 MHz/0 flash wait-states (0 WS). The Cortex-M4F target provides 640 KB static random-access memory (SRAM) and 2 MB flash. All values are cycle counts and use the selected backends in Table 1 unless stated otherwise.

4.1 Hybrid KEM Results

Table 2 reports the main Hash-CT X25519 benchmark configuration with $N = 100$ medians. ML-KEM, NTRU+, and SMAUG-T totals for key generation, encapsulation, and decapsulation are 3,719,838–8,770,987 cycles, whereas HQC totals are 19,637,683–74,467,370 cycles with the v5-compatible additive-FFT backend [7]. These totals include ct_{pq} hashing in the benchmark transcript, so large-ciphertext rows such as HQC include an additional Hash-CT hashing cost relative to a no-ct_{pq} transcript. Thus the selected KEM rows split into two latency bands in this stack, with lattice-based hybrids below 8.8 M cycles and HQC hybrids above 19.6 M cycles.

Table 2: Hash-CT hybrid KEM benchmark rows on Cortex-M4 at 20 MHz/0 WS (X25519), $N = 100$ medians. Cycle counts include ct_{pq} hashing. HQC uses the v5-compatible additive-FFT backend.

PQ-KEM	KeyGen	Encaps	Decaps	Total
SMAUG-T1	1,210,134	1,820,980	1,412,908	4,444,022
SMAUG-T3	1,836,280	2,439,134	2,128,255	6,403,669
SMAUG-T5	2,606,616	3,189,011	2,975,360	8,770,987
NTRU+768	1,043,984	1,646,596	1,054,290	3,744,870
NTRU+864	1,156,249	1,753,290	1,175,628	4,085,167
NTRU+1152	1,314,452	1,910,138	1,342,069	4,566,659
ML-KEM-512	1,010,490	1,643,740	1,065,608	3,719,838
ML-KEM-768	1,258,669	1,909,217	1,342,781	4,510,667
ML-KEM-1024	1,635,681	2,281,523	1,728,766	5,645,970
HQC-128	3,596,009	6,536,578	9,505,096	19,637,683
HQC-192	8,854,451	16,036,945	23,970,881	48,862,277
HQC-256	13,802,173	24,020,196	36,645,001	74,467,370

As sanity checks, local standalone anchors measured X25519 key generation, encapsulation, and decapsulation at 0.616/1.230/0.615 M cycles, and Ed25519 key generation, signing, and verification at 5.94/11.92/5.99 M cycles. Replacing X25519 with P-256 changes all twelve KEM totals by at most 0.347% (per-row ratios stay within 0.997–1.000×) and does not change the family ordering. Deterministic concat-to-nested checks change representative signature rows by at most 0.35% and guard backend selection and wrapper wiring.

4.2 Hybrid DSA Results and Tail Behavior

Signature measurements use the Ed25519 concat baseline unless stated otherwise. Table 3 gives the $N = 100$ lattice/FN-DSA rows used for the main comparison, while Table 4 keeps low-repetition AIMer/SPHINCS$^+$ scale-only rows separate. ML-DSA has the lowest measured lattice-signature signing latency, HAETAE shows rejection-sampling variability, and FN-DSA combines fast verification with expensive key generation. Because several key-generation and signing paths are randomized, Table 5 reports tail summaries for the lattice/FN-DSA set instead of relying only on medians.

Wall-clock interpretation. At the measured 20 MHz setting, the KEM latency bands in Table 2 correspond to approximately 0.19–0.44 s for lattice-based hybrids and 0.98–3.72 s for HQC hybrids. Thus the difference is not only a cycle-

Table 3: Main hybrid DSA comparison rows (PQ + Ed25519, concat) on Cortex-M4 at 20 MHz/0 WS. Cycle counts. All rows are $N = 100$ medians.

PQ-DSA	KeyGen	Sign	Verify
ML-DSA-44	7,364,995	15,037,912	7,563,684
ML-DSA-65	8,460,040	17,092,257	8,559,121
ML-DSA-87	10,220,574	18,087,747	10,316,385
HAETAE2	12,028,569	31,801,353	6,943,470
HAETAE3	15,915,932	36,271,144	7,682,232
HAETAE5	19,076,721	41,939,790	8,116,242
FN-DSA-512	75,087,719	34,401,102	6,545,554
FN-DSA-1024	263,700,870	60,320,802	6,921,437

Table 4: Scale-only low-repetition hybrid DSA rows (PQ + Ed25519, concat) on Cortex-M4 at 20 MHz/0 WS. Cycle counts. AIMer192f/256s are $N = 10$ medians. AIMer128f is an upstream `m4opt` single-run measurement ($N = 1$) included to align with Table 8. SPHINCS$^+$ rows are $N = 5$ medians. These rows are not used for ranking, variance, or distributional conclusions.

PQ-DSA	KeyGen	Sign	Verify
AIMer128f (m4opt)	6,404,117	47,935,060	32,960,014
AIMer192f (m4fast)	7,181,956	89,527,934	78,413,565
AIMer256s (m4stack)	10,190,847	1,971,649,069	1,550,023,748
SPHINCS$^+$-SHA2-128f	20,670,562	356,674,443	27,764,750
SPHINCS$^+$-SHA2-128s	948,458,267	7,170,561,799	13,657,848
SPHINCS$^+$-SHA2-256f	64,504,322	1,309,374,104	42,139,178
SPHINCS$^+$-SHAKE-128f	56,181,392	1,188,091,067	75,088,359
SPHINCS$^+$-SHAKE-256f	204,948,947	4,015,467,171	114,616,346
SPHINCS$^+$-SHAKE-192s	4,774,330,073	42,882,388,698	41,997,450

count ordering, because it changes the deployment regime from sub-second local key establishment to multi-second operation on the same Cortex-M4 stack. Signature rows show a similar split. ML-DSA signing takes about 0.75–0.90 s, HAETAE signing about 1.59–2.10 s, FN-DSA-1024 key generation about 13.2 s, AIMer256s signing about 98.6 s, and the slowest SPHINCS$^+$-SHAKE signing rows reach tens of minutes at 20 MHz.

Tail implication. The non-obvious point is not merely that rejection sampling has tails, but that the deployment ordering depends on the statistic used as the service objective. In the nominal level-5 signing rows, HAETAE5 is faster than FN-DSA-1024 at the median (41.94 M vs. 60.32 M cycles) and still has the lower mean (51.23 M vs. 60.33 M), but the retained logs show an observed HAETAE5 maximum of 189.8 M cycles (4.53× its median), whereas FN-DSA-1024 stays at 60.58 M cycles (about 1.00× its median). Thus a throughput-oriented deployment and a deadline-sensitive deployment can read the same median table differently. Across the lattice/FN-DSA rows, the signing-jitter spectrum is also visible. FN-DSA is essentially flat in this campaign (1.0× max/min), ML-DSA has moderate spread (2.0–2.2×), and HAETAE has heavy spread (5.8–9.1×). These summaries are not percentile or worst-case execution-time estimates. They are latency service-objective observations under this harness.

Table 5: Key-generation and signing tail summary at 20 MHz/0 WS (Ed25519, concat), $N = 100$ runs. Medians are M cycles. Mean columns are mean/median and range columns are max/min from retained logs.

Scheme	KeyGen med.	KeyGen mean	KeyGen range	Sign med.	Sign mean	Sign range
HAETAE2	12.03	1.15×	5.1×	31.80	1.17×	8.9×
HAETAE3	15.92	1.13×	5.1×	36.27	1.14×	5.8×
HAETAE5	19.08	1.21×	10.6×	41.94	1.22×	9.1×
ML-DSA-44	7.36	1.00×	1.0×	15.04	1.05×	2.0×
ML-DSA-65	8.46	1.00×	1.0×	17.09	1.05×	2.2×
ML-DSA-87	10.22	1.00×	1.0×	18.09	1.07×	2.0×
FN-DSA-512	75.09	1.09×	3.3×	34.40	1.00×	1.0×
FN-DSA-1024	263.70	1.19×	4.2×	60.32	1.00×	1.0×

4.3 Keccak and Hash Backend Effects

The Keccak experiment changes only Keccak-$f[1600]$. Table 6 lists representative exact rows from the Keccak-isolation campaign. Control rows (SPHINCS$^+$-SHA2 and FN-DSA) remain at 1.00×, supporting attribution to the Keccak permutation rather than unrelated build or measurement noise. The largest exact effect is SPHINCS$^+$-SHAKE-256f signing, which changes from 8,663,447,709 to 4,015,467,171 cycles (2.16×). The largest AIMer row is AIMer192s (m4stack) signing at 1.65×. Lattice rows gain 1.18–1.51×, and HQC-128 gains 1.10×. A shared symmetric backend therefore changes SHAKE-heavy deployment conclusions more than arithmetic-heavy KEM conclusions.

Table 6: Keccak-$f[1600]$ backend attribution with representative rows from the Keccak-isolation campaign. Only the permutation changes. All higher-level SHAKE/SHA3 code, arithmetic kernels, classical code, target configuration, and harness stay fixed. Medians may differ slightly from the primary campaign rows in Table 2. Speedup = C-reference/ARM assembly.

Class	Row	C-ref	ASM	Speedup
Lattice KEM	ML-KEM-1024 total	8,513,176	5,645,764	1.51×
HQC	HQC-128 total	21,575,619	19,647,266	1.10×
ML-DSA	ML-DSA-87 sign	23,042,134	18,087,747	1.27×
HAETAE	HAETAE5 sign	63,096,274	41,939,790	1.50×
AIMer	AIMer192s (m4stack) sign	1,541,036,168	934,921,177	1.65×
SPHINCS$^+$-SHAKE	SPHINCS$^+$-SHAKE-256f sign	8,663,447,709	4,015,467,171	2.16×
Control row	SPHINCS$^+$-SHA2-128f sign	356,601,179	356,674,443	1.00×
Control row	FN-DSA-512 sign	34,392,093	34,401,102	1.00×

4.4 SMAUG-T Local Backend and Optimization Dilution

Table 7 isolates the local SMAUG-T arithmetic backend. With C-Keccak, fixed harness, and fixed test-vector streams, standalone SMAUG-T improves by 1.79–1.82× over the portable C polynomial-multiplication path. In SMAUG-T+X25519, the speedup drops to 1.42–1.65× because non-PQ work remains fixed. The residual column reports Hybrid ASM minus Stand ASM in the C-Keccak isolation run. About 2.56 M cycles remain for X25519, Hash-CT/QSF hashing, wrapper work, and harness overhead. Normalized by the primary

Table 7: SMAUG-T local polynomial-multiplication backend. Standalone/Hybrid C and ASM columns are from the within-experiment C-Keccak isolation run. Residual is Hybrid ASM − Stand ASM in M cycles. Residual share normalizes that residual by the primary assembly-Keccak X25519 hybrid total.

Scheme	Stand C	Stand ASM	Hybrid C	Hybrid ASM	Hyb. spd.	Res. M cyc.	Res. share
SMAUG-T1	5,093,516	2,811,333	7,653,239	5,371,074	1.42×	2.56	58%
SMAUG-T3	10,352,582	5,788,205	12,912,889	8,348,500	1.55×	2.56	40%
SMAUG-T5	16,834,070	9,226,662	19,395,083	11,787,708	1.65×	2.56	29%

assembly-Keccak X25519 hybrid totals, this residual is 58%, 40%, and 29% for T1/T3/T5.

Cross-paper SMAUG-T numbers are contextual. Narlı et al. [22] report 3,239,225, 5,478,853, and 9,123,609 cycles for T1/T3/T5, while our primary standalone assembly totals are 1,936,585, 3,895,476, and 6,262,304 cycles. Because board, toolchain, and wait-state details may differ, Table 7's within-experiment C-to-ASM ratios are the quantitative comparison used here.

Deployment rule. On this Cortex-M4 stack, an engineer comparing SHAKE-heavy signatures should first validate the shared Keccak backend. For SMAUG-T hybrids, standalone arithmetic gains should be read through the residual share in Table 7. Small T1 rows become residual-bound sooner, whereas T5 exposes more PQ arithmetic improvement in the composed total.

4.5 Size and Memory Considerations

Table 8 combines wire size, peak-stack watermark, incremental scheme-owned code estimate, and representative linked harness footprint. The largest representative stack is HAETAE2+Ed25519 signing at 83,468 B, and the largest KEM watermark shown is HQC-128+X25519 decapsulation at 53,600 B. On this 640 KB SRAM target, the measured rows are limited earlier by latency, wire size, or flash footprint than by stack capacity.

As a backend sanity check (not an optimized ECDSA-vs-Ed25519 comparison), substituting ECDSA-P256 for Ed25519 in the hybrid lattice/FN-DSA rows makes generic mbedTLS-C ECDSA-P256 verification dominate every total at 122–126 M cycles ($N = 100$ medians, 20 MHz/0 WS), swamping all post-quantum signing and key-generation differences. This is an artifact of generic-C P-256 arithmetic, not a recommendation to prefer one classical signature backend.

Limitations. The campaign uses one NUCLEO-L4R5ZI board, one selected Cortex-M4 backend stack, and the 20 MHz/0 WS setting. It measures primitive/harness-layer timing and footprint, not TLS/X.509 integration, energy, side-channel leakage, fault behavior, formal constant-time proofs, or worst-case execution-time bounds. The Hash-CT KEM rows include ct_{pq} hashing under this benchmark transcript and should not be read as IETF X-Wing deployment costs.

Table 8: Deployment footprint of representative hybrids. Code est. is an incremental object-sum excluding shared Keccak/HAL/harness. The final line reports representative full linked memory-harness Executable and Linkable Format (ELF) flash/random-access memory (RAM) from `arm-none-eabi-size`, not a single all-algorithm deployment binary.

Hybrid	Type	pk	ct/sig	Peak stack	Code est.
ML-KEM-768 + X25519	KEM	1,216	1,120	6,748	20,460
SMAUG-T3 + X25519	KEM	1,120	1,024	18,336	15,988
HQC-128 + X25519	KEM	2,273	4,465	53,600	216,335
ML-DSA-44 + Ed25519	DSA	1,344	2,488	45,216	29,288
HAETAE2 + Ed25519	DSA	1,024	1,542	83,468	39,048
AIMer128f + Ed25519	DSA	64	5,956	11,952	22,820
SPHINCS$^+$-SHAKE-128f + Ed25519	DSA	64	17,156	18,860	13,880

Full linked harness flash/RAM (text+data / data+bss) is 1,024,416/12,312 B for KEM rows and 643,484/23,852 B for DSA rows.

5 Conclusion

This paper measured KpqC/NIST hybrid KEM benchmark rows with X25519, following the QSF component interface under a uniform Hash-CT transcript, together with Bindel et al. hybrid signature AND-combiner rows, on ARM Cortex-M4 under one bare-metal harness. The aim was composed-cost attribution, not backend-independent primitive ranking, measurement of the IETF X-Wing wire transcript, or security claims about new constructions.

The main lesson is that composed hybrid benchmarks behave differently from primitive-only rankings. A shared symmetric backend can dominate SHAKE-heavy signatures, median-only signing tables can hide deadline-relevant tails, and standalone arithmetic optimizations can be partially absorbed by fixed classical, hashing, and wrapper work. These effects change what an implementation engineer should check first, namely backend provenance, tail summaries for randomized operations, and end-to-end hybrid measurements for any claimed optimization.

On the selected Cortex-M4 stack, the measured rows separate lattice KEM hybrids and HQC hybrids into different latency regimes, but the absolute totals are transcript- and backend-specific. Hybrid PQC reports should therefore fix shared backend provenance, report randomized-operation tails, and evaluate optimizations inside composed hybrids. These are timing and integration findings, not side-channel or construction-security claims.

References

1. Adomnicai, A.: An update on keccak performance on armv7-m. Cryptology ePrint Archive (2023)
2. Barbosa, M., Connolly, D., Duarte, J.D., Kaiser, A., Schwabe, P., Varner, K., Westerbaan, B.: X-wing: The hybrid kem you've been looking for. Cryptology ePrint Archive (2024)
3. Bernstein, D.J., Van Gastel, B., Janssen, W., Lange, T., Schwabe, P., Smetsers, S.: Tweetnacl: A crypto library in 100 tweets. In: International Conference on Cryptology and Information Security in Latin America. pp. 64–83. Springer (2014)

4. Bindel, N., Herath, U., McKague, M., Stebila, D.: Transitioning to a quantum-resistant public key infrastructure. In: International Workshop on Post-Quantum Cryptography. pp. 384–405. Springer (2017)
5. Bos, J.W., Costello, C., Naehrig, M., Stebila, D.: Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In: 2015 IEEE symposium on security and privacy. pp. 553–570. IEEE (2015)
6. Bürstinghaus-Steinbach, K., Krauß, C., Niederhagen, R., Schneider, M.: Post-quantum tls on embedded systems: Integrating and evaluating kyber and sphincs+ with mbed tls. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. pp. 841–852 (2020)
7. Chen, M.S., Chien, T.Y., Chiu, C.M., Lin, H.H., Peng, C.T., Yang, B.Y.: Additive ffts for hqc on arm cortex-m4, revisited. Cryptology ePrint Archive (2026)
8. Cheon, J.H., Choe, H., Devevey, J., Güneysu, T., Hong, D., Krausz, M., Land, G., Möller, M., Stehlé, D., Yi, M.: Haetae: Shorter lattice-based fiat-shamir signatures. IACR Transactions on Cryptographic Hardware and Embedded Systems **2024**(3), 25–75 (2024)
9. Cheon, J.H., Choe, H., Seo, J., Seong, H.: Smaug (-t), revisited: Timing-secure, more compact, less failure. IEEE Access **12**, 188386–188397 (2024)
10. Cho, S.: NTRU+ KEM on ARM Cortex-M4: Optimized implementation and benchmarks. GitHub repository, https://github.com/SuBeen-Cho/Optimization_CortexM4_NTRUplus (2026)
11. Connolly, D., Hövelmanns, K., Hülsing, A., Kousidis, S., Meijers, M.: Starfighters—on the general applicability of x-wing. Cryptology ePrint Archive (2025)
12. Connolly, D., Schwabe, P., Westerbaan, B.: X-wing: General-purpose hybrid post-quantum kem. Internet-Draft draft-connolly-cfrg-xwing-kem-10, Internet Engineering Task Force (Mar 2026), https://datatracker.ietf.org/doc/draft-connolly-cfrg-xwing-kem/, version 10, 2 March 2026
13. Driscoll, F., Parsons, M., Hale, B.: Terminology for post-quantum traditional hybrid schemes. RFC 9794, IETF (2025)
14. Gonzalez, R., Wiggers, T.: Kemtls vs. post-quantum tls: Performance on embedded systems. In: International Conference on Security, Privacy, and Applied Cryptography Engineering. pp. 99–117. Springer (2022)
15. Kannwischer, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: pqm4: Testing and benchmarking nist pqc on arm cortex-m4. NIST Second Post-Quantum Cryptography Standardization Conference (2019)
16. KpqC-cryptocraft: KpqClean Ver2: Reference implementations of the KpqC standardized algorithms. GitHub repository, https://github.com/kpqc-cryptocraft/KpqClean_ver2 (2026)
17. KpqC Organizing Committee: Selected algorithms from the KpqC competition round 2. https://kpqc.or.kr/competition_02.html (2025), final algorithms announced 16 January 2025, accessed 2026-06-26
18. Kwon, J., Lee, S., Lee, B., Seo, H., Cho, J.: Efficient implementations of aimer post-quantum signature scheme for low-end to high-end iot devices. IEEE Internet of Things Journal (2025)
19. Lenngren, E.: P256-Cortex-M4: P-256 ECDSA/ECDH for Cortex-M4. https://github.com/Emill/P256-Cortex-M4
20. Lenngren, E.: X25519-Cortex-M4. https://github.com/Emill/X25519-Cortex-M4
21. Mbed TLS Team: Mbed TLS. GitHub repository, https://github.com/Mbed-TLS/mbedtls (2024)

22. Narlı, O., Pehlivanoğlu, M.K., Akleylek, S.: A new optimized implementation of smaug-t for lightweight devices. In: Nordic Conference on Secure IT Systems. pp. 77–90. Springer (2025)
23. National Institute of Standards and Technology: FIPS 203: Module-lattice-based key-encapsulation mechanism standard (ML-KEM). Federal information processing standards publication, NIST (Aug 2024)
24. National Institute of Standards and Technology: FIPS 204: Module-lattice-based digital signature standard (ML-DSA). Federal information processing standards publication, NIST (Aug 2024)
25. National Institute of Standards and Technology: FIPS 205: Stateless hash-based digital signature standard (SLH-DSA). Federal information processing standards publication, NIST (Aug 2024)
26. National Institute of Standards and Technology: NIST IR 8545: Status report on the fourth round of the NIST post-quantum cryptography standardization process. Tech. rep., NIST (Mar 2025), HQC selected for standardization
27. National Institute of Standards and Technology: Recommendations for key-encapsulation mechanisms. NIST Special Publication 800-227, NIST (Sep 2025). https://doi.org/10.6028/NIST.SP.800-227
28. Perlner, R.: FIPS 206 status update: FN-DSA (Falcon). Presentation, 6th NIST PQC Standardization Conference (2025), nIST
29. Pornin, T.: Falcon on ARM Cortex-M4: An update. Cryptology ePrint Archive, Report 2025/123 (2025)
30. PQClean Contributors: PQClean: Clean, portable, tested implementations of post-quantum cryptography. https://github.com/PQClean/PQClean
31. Sim, M., Song, G., Eum, S., Lee, M., Yoon, S., Baksi, A., Seo, H.: Integrating and benchmarking kpqc in tls/x. 509. Electronics **14**(18), 3717 (2025)
32. Weatherley, R.: ARMv7-M SHA-256 assembly. GitHub repository, https://github.com/rweather/lwc-finalists
33. X-Wing KEM Team: X-Wing C implementation. GitHub repository, https://github.com/x-wing-kem-team/xwing (2024), commit `c696a197`, accessed 2026-06-27