computers

MDPI

Article

Auditable Conformance and Cross-Library Interoperability Testing for ML-KEM and ML-DSA

Sijiang Xie *, Xingyu Lu, Haida Wang and Hong Zhao

Department of Cyberspace Security, Beijing Electronic Science and Technology Institute, Beijing 100070, China
* Correspondence: xiesj@besti.edu.cn

Abstract

An implementation that passes internal round-trip tests may still produce artifacts that another library cannot consume. We present an auditable workflow for selected-vector conformance and raw-artifact interoperability testing of ML-KEM and ML-DSA in liboqs and PQMagic. The workflow links FIPS 203/204 Automated Cryptographic Validation Protocol (ACVP) projections, bidirectional producer–consumer cases, and deterministic negative tests to vector and binary hashes. For pinned liboqs 0.15.0 and PQMagic-SHAKE builds, both adapters matched all 690 expected implementation-vector outcomes across ML-KEM-512/768/1024 and ML-DSA-44/65/87. All 1200 bidirectional positive cases and 90 deliberately sparse deterministic one-bit mutation cases produced their specified outcomes. A supporting regression check passed 12 persisted-artifact replay cases between two PQMagic builds from the same commit; it does not establish cross-version compatibility. Performance measurements provide a descriptive single-session snapshot on one Windows host, with 50 timed batch observations per operation. Aigis and SHAKE/SM3 comparisons are supporting configuration analyses, not extensions of the standardized cross-library compatibility claim. The accompanying deployment considerations are a taxonomy and checklist, not a validated decision procedure. The contribution is an auditable record of specified artifact-level behavior, bounded by the selected vectors, recorded commits, host, and interfaces. It does not establish product certification, protocol interoperability, or stable performance rankings across sessions.

Keywords: post-quantum cryptography; ML-KEM; ML-DSA; interoperability; conformance testing; auditable evaluation; liboqs; PQMagic

Check for updates

Academic Editor: Paolo Bellavista

Received: 27 August 2026
Revised: 11 September 2026
Accepted: 18 September 2026
Published: 22 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Post-quantum deployment requires more than selecting a standardized algorithm [1–4]. FIPS 203 and FIPS 204 define ML-KEM and ML-DSA [5,6], but an implementation change can still affect encoded keys, ciphertexts, signatures, contexts, and stored artifacts. Operation-level benchmarks quantify cost, and internal round trips establish self-consistency. Neither test shows that two library distributions consume each other's standardized bytes.

Prior evaluations address complementary evidence layers. Abbasi et al. compare post-quantum performance across heterogeneous platforms [7], while Souvatzidaki and Limniotis and Raavi et al. examine TLS 1.3, PKI, and signature integration [8,9]. Recent deep-learning studies include a visual–textual mutual guidance fusion network for remote-sensing visual question answering [10] and a graph-representation-learning plot-to-track association framework for compact HFSWR [11]. Recent PQC surveys and deployment

studies further summarize algorithm families, migration challenges, constrained-device benchmarking, cryptographic-library support, and TLS benchmarking [12–16]. These works provide current context; they are not pooled with the present measurements. The specific gap addressed in this paper is an auditable connection between selected standard-vector checks, cross-library producer–consumer tests, and negative behavior. Performance benchmarking and persisted-artifact replay provide supporting context rather than separate claims of general deployment readiness.

We address this gap with a dependency-ordered evidence pipeline for ML-KEM and ML-DSA. Selected ACVP projections test expected FIPS 203/204 outputs, and bidirectional cases test whether each library consumes artifacts produced by the other. Deterministic mutations test verification failure or the specified implicit-rejection behavior. The pipeline preserves failed and not-applicable cases and binds results to vector and binary hashes. An additional same-commit replay check uses isolated producer and consumer processes to test persisted artifacts across two builds, without testing release evolution.

This work addresses two primary research questions. RQ1 asks whether the pinned liboqs and PQMagic-SHAKE adapters reproduce the expected outcomes of selected FIPS 203/204 ACVP projections. RQ2 asks whether the two distributions consume each other's raw ML-KEM and ML-DSA artifacts across all six parameter sets and satisfy the specified negative-case outcomes. The timing, Aigis, and SHAKE/SM3 analyses describe the tested configurations and their measurement boundaries. Same-commit replay is a supporting artifact-regression check, not a separate claim about version-to-version compatibility.

The primary contribution is an executed, auditable conformance and cross-library interoperability workflow for all three ML-KEM and three ML-DSA parameter sets. It separates exact vector outcomes, producer–consumer compatibility, and deterministic negative behavior instead of inferring compatibility from successful self-tests. The same-commit replay check is a minor corroborating result. Adaptive-batch timing, Aigis measurements, and SHAKE/SM3 comparisons provide supporting observations rather than coequal contributions. Section 5 organizes deployment considerations as a taxonomy and checklist; it does not propose a validated selection method. The conclusions concern the pinned distributions and tested raw interfaces, not general substitutability.

The scope is deliberately limited to one Windows x86_64 host, selected vectors, raw cryptographic artifacts, and recorded commits. It does not evaluate protocol negotiation, X.509, TLS, QUIC, SPDM, ACME, GPU, or energy behavior, native-memory use, or side channels. PQMagic-SM3 and Aigis are reported as separate configurations because liboqs provides no aligned counterpart in this experiment. The replay result compares two builds from the same PQMagic commit and is not evidence of compatibility across software versions.

2. Cryptographic and Engineering Background

2.1. Module-Lattice Foundations and Standardized Algorithms

Learning with Errors (LWE) asks an adversary to recover a secret from noisy linear relations and provides a foundation for modern lattice cryptography [17]. Ring-LWE and module variants introduce algebraic structure that enables compact keys and fast polynomial arithmetic [18]. For the parameter sets considered here, computation takes place over a quotient ring such as

$$R_q = \mathbb{Z}_q[X]/(X^n + 1). \quad (1)$$

A simplified module-LWE public relation is

$$b = As + e(\mathrm{mod} q), \tag{2}$$

where A is public, s is short and secret, and e is sampled from a narrow error distribution. Security is not inferred merely from the presence of noise; concrete hardness depends on dimensions, modulus, distributions, attacks, and cost models [19]. This is why NIST categories should be reported as categories rather than silently replaced by a continuous 128/192/256-bit axis.

ML-KEM is the standardized module-lattice KEM in FIPS 203 [5]. Its lineage includes the Kyber construction and a Fujisaki–Okamoto-style conversion from an encryption core to chosen-ciphertext security [20,21]. Modern analyses refine the assumptions and random-oracle treatment of such conversions [22,23]. A KEM exposes key generation, encapsulation, and decapsulation. In deployment terms, these operations need not occur equally often: a server may reuse a configured key for many encapsulations, while a client or service may perform decapsulation on every connection. Therefore, summing the three means is useful only as a compact descriptive value, not as a universal workload model.

ML-DSA is standardized by FIPS 204 and derives from Dilithium [6,24]. It combines module-lattice relations with Fiat–Shamir-style challenges, decomposition, and rejection sampling. Rejection sampling makes signing time intrinsically variable because candidate responses may be discarded. A benchmark that reports only a minimum or a single successful signature can therefore favor an unusually short path. Verification follows a different arithmetic and hashing path, so a signing advantage does not guarantee a verification advantage.

SHAKE128 and SHAKE256 are extendable-output functions standardized in FIPS 202 [25]. ML-KEM and ML-DSA use Keccak-derived functionality for expansion, hashing, and sampling. PQMagic can substitute SM3 in supported configurations. SM3 is standardized in GB/T 32905-2016 [26]. Substitution does not have a constant cost because different operations invoke hash and extendable-output functionality with different message lengths, call counts, and surrounding arithmetic. The correct experimental object is thus the algorithm-operation-backend tuple rather than the library name alone.

Table 1 defines the evidence vocabulary used throughout this paper. This vocabulary prevents a common reporting failure in which a modeled certificate size, a measured public-key byte length, and an asymptotic expression appear in the same table without provenance.

Table 1. Evidence types and permitted interpretations.

Evidence Type	Definition	Permitted Inference	Prohibited Inference
Measured	Observed by a stated benchmark with an identified scope	Performance or size under that scope	Universal algorithm ranking
Derived	Calculated from measured inputs	Ratios, throughput, and confidence intervals	Independent physical measurement
Estimated	Produced by an explicit model	Scenario sizing under stated assumptions	Claim of generated protocol artifacts
Theoretical	Obtained from standards or primary literature	Complexity, hardness assumptions, and security categories	Observed runtime or proof of implementation security

2.2. Aigis Constructions and Domestic Implementation Context

Aigis-Enc and Aigis-Sig are asymmetric module-lattice constructions that adjust distributions and rounding choices to alter the balance between public and secret computations [27]. Here, they provide supporting context for additional configurations exposed by PQMagic. They are outside the primary ML-KEM/ML-DSA conformance and cross-library interoperability questions because no aligned liboqs counterpart was tested. The measurements do not establish new security reductions. Security statements remain attributed to the source construction; this study reports only the tested implementation's functional behavior, timing, and exposed artifact sizes.

For Aigis-Enc, an asymmetric module-LWE sample may be summarized as Equation (2) with distinct secret and error distributions. A compressed public component can be written as

$$\tilde{b} = Compress_{q \to p_t}(As + e). \tag{3}$$

Correct decryption depends on the aggregate noise, including compression error. A schematic error term is

$$w = e^T r - s^T e_1 + e_2 - \epsilon_v - \epsilon_t^T r + s^T \epsilon_u. \tag{4}$$

The infinity norm $\| \quad \|_\infty$ must remain below the decoding threshold. Omitting compression terms is acceptable for a high-level LWE explanation but not for a complete correctness argument. Aigis-Enc then applies an FO-style transformation so that encapsulation derives coins and keying material from the message, public key, and ciphertext:

$$(\hat{K}, r) = G(m \parallel H(pk)), c = Enc(pk, m; r), K = KDF(\hat{K} \parallel H(c)). \tag{5}$$

Aigis-Sig combines AMLWE-style hiding with an asymmetric module short-integer-solution relation. In simplified form, the adversarial objective includes finding a nonzero short vector z satisfying

$$Az = \mathbf{0} (\mathrm{mod} q), 0 < \|z\|_\infty \leq \beta. \tag{6}$$

The implemented signature includes commitment generation, challenge derivation, response formation, decomposition, hints, and rejection sampling. A successful functional test demonstrates that a generated signature verifies and that a modified message is rejected; it does not demonstrate strong unforgeability, validate constant-time behavior, or transfer every theorem from a publication to every later code revision. That separation is retained in the deployment checklist.

2.3. Algorithm Families and Engineering Consequences

Table 2 summarizes the schemes that appear either in the aligned comparison or in the broader baseline. The complexity entries are theoretical evidence. They communicate scaling and dominant operations, not measured instruction counts. An expression such as $O(n \log n)$ does not determine a runtime constant, cache behavior, vectorization efficiency, or library-call overhead.

Classic McEliece illustrates why primitive latency and artifact size must be considered together. Code-based decapsulation may be acceptable in an environment where a large public key is provisioned once, while transmitting that public key in every session may be unacceptable. Optimized embedded work confirms that platform-aware implementation matters even for a construction with a long cryptanalytic history [28]. FrodoKEM provides a different trade-off: its plain-LWE structure is conservative, but dense matrix arithmetic and large artifacts increase computational and network cost. Falcon offers compact signatures,

but its sampling and implementation-assurance requirements differ from the comparatively regular ML-DSA structure. FIPS 205 adds a standardized hash-based alternative whose security diversity is valuable, although SLH-DSA was not timed in this experiment [29].

Table 2. Security foundations, dominant operations, and engineering implications.

Scheme/Family	Security Basis	Dominant Software Work	Indicative Complexity	Engineering Implication
ML-KEM	Module-LWE with FO-style CCA conversion	NTT, polynomial arithmetic, Keccak	$O(k^2 n \log n)$	Balanced standardized KEM; moderate artifacts
ML-DSA	Module-LWE and Module-SIS	NTT, rejection sampling, hashing	Expected $O(k^2 n \log n)$	Fast verification; signatures larger than ECC
Aigis-Enc	AMLWE/AMLWE-R with FO transformation	Compressed module-lattice arithmetic	$O(k^2 n \log n)$	Domestic implementation path; interoperability must be managed
Aigis-Sig	AMLWE and AMSIS	NTT, decomposition, rejection sampling	Expected $O(k^2 n \log n)$	Domestic signature option with implementation-specific evidence
Classic McEliece	Binary Goppa-code syndrome decoding	Finite-field operations and decoding	Parameter dependent	Very large public key but small ciphertext
FrodoKEM	Plain LWE	Dense matrix arithmetic	$O(n^2)$	Conservative structure with high bandwidth cost
Falcon/FN-DSA	NTRU lattices	FFT-like sampling	$O(n \log n)$	Compact signature but demanding constant-time implementation

2.4. Related Benchmarking and Interoperability Evidence

liboqs and PQMagic provide overlapping standardized algorithms but differ in ecosystem, optional backends, and additional mechanisms [30–32]. A same-host comparison reduces hardware variation only when algorithm names, parameter sets, operation semantics, and timing boundaries are aligned. Aigis and SM3 configurations are therefore evaluated separately rather than treated as substitutes for standardized liboqs interfaces.

Performance studies establish the importance of platform and operation choice. Abbasi et al. measured several post-quantum algorithms across server, laptop, and constrained environments [7], while implementation studies on embedded targets show that platform-aware optimization remains decisive [28]. These results are valuable baselines, but differences in versions, compilers, processors, and timing loops prevent causal speedup claims across unmatched systems.

Backend and algorithm comparisons provide supporting measurement context. A single library-wide multiplier is not operationally useful when workloads invoke key generation, encapsulation, decapsulation, signing, and verification at different frequencies. The supporting analysis therefore retains the algorithm-operation-backend tuple and reports Aigis and SM3 configurations within their own standards and policy boundaries. These observations do not broaden the standardized cross-library interoperability claim.

Protocol studies address a different layer. Souvatzidaki and Limniotis show that packet loss and handshake size affect TLS 1.3 latency [8], while Raavi et al. examine signature choices in TLS and PKI [9]. Those results motivate artifact-size and integration analyses but

cannot be inferred from native-call timings. Conversely, protocol success with one library does not establish exact cross-library behavior for keys, ciphertexts, contexts, or signatures.

Conformance and raw-artifact interoperability are the primary focus. A self-test can pass with a mutually consistent but non-standard encoding. Selected official vectors test expected bytes, whereas cross-library producer–consumer cases test the specified artifact boundary. The paper does not claim that interoperability testing itself is new. It contributes an auditable combination of selected ACVP projections, full-parameter bidirectional liboqs/PQMagic cases, deterministic negative tests, and binary provenance. The same-commit replay check is retained as minor corroboration of persisted-artifact handling.

3. Experimental Methodology

3.1. Experimental Scope and Platform

The experiment was performed on one Windows 10 x86_64 host, identified by CPU family and model metadata in Table 3. This controlled same-host comparison reduces hardware variation but does not support cross-platform performance claims. Instruction-set entries indicate host capability only; the experiment did not trace which optimized path each implementation executed. Linux, ARM, and RISC-V were not tested, so scheduling, ABI, compiler, and architecture effects cannot be separated from the recorded timings. Reproduction therefore requires the complete host, OS, compiler, build-option, binary-hash, and runtime-dispatch record.

Table 3. Platform and build configuration.

Item	Configuration	Evidence Role
Host	Windows 10, x86_64, Intel Family 6 Model 141	Single host; Microsoft/Redmond WA and Intel/Santa Clara CA, USA
Python	3.13.0; PSF (Beaverton, OR, USA)	Benchmark orchestration only
PQMagic build	PQMagic/PQCrypto 9613aa3c; Release; Clang 20.1.2; x86_64	LLVM/llvm.org; Kitware/Clifton Park NY, USA; SHAKE/SM3 builds
liboqs runtime	liboqs/OQS 0.15.0 (97f6b86b); 29 KEMs and 221 signature mechanisms	OQS/openquantumsafe.org; common ML-KEM/ML-DSA baseline
Instruction availability	SSE2/SSE4, AES, PCLMULQDQ, AVX/AVX2/AVX-512, SHA reported available	Capability metadata; not proof of per-path instruction use
Repeated observations	50 adaptive batches per operation after 10 warm-ups	Repeated batches within the primary session
Batch target	Approximately 20 ms per timed batch	Reduces timer and Python-call granularity

PQMagic was built in two independent release configurations. One enabled SHAKE and disabled SM3; the other enabled SM3 and disabled SHAKE. Both exposed ML-KEM, ML-DSA, Aigis-Enc, and Aigis-Sig. liboqs supplied the aligned ML-KEM and ML-DSA mechanisms. The broad baseline additionally measured X25519, ECDH P-256, RSA, Classic McEliece, FrodoKEM, ECDSA P-256, Ed25519, and Falcon. The baseline provides deployment context; it is not used to claim that unlike primitives provide identical protocol semantics.

3.2. Correctness Gates

Performance data were admitted only after functional checks. For every KEM, the benchmark generated a key pair, encapsulated a shared secret, decapsulated the ciphertext, and required exact equality of the shared secrets. For every signature, it generated a key pair, signed a fixed message, required successful verification, changed the message, and

required rejection. Thirty-two correctness rows passed. Representative native benchmark executables were then run for SHAKE and SM3 configurations, producing eight successful return-code checks. These native checks are corroboration rather than a second statistical sample because their internal loops and outputs differ from the unified timer. Table 4 summarizes the functional gates, calibration, measurement, and recorded outputs.

Table 4. Correctness, calibration, and measurement protocol.

Stage	KEM Gate	Signature Gate	Recorded Output
Positive correctness	Encapsulated and decapsulated secrets must match	Valid signature must verify	Pass/fail status
Negative correctness	Not applicable to the selected API gate	Modified message must be rejected	Return code and rejection status
Warm-up	10 untimed calls	10 untimed calls	Warm-up count
Calibration	Batch size doubled to about 20 ms	Same procedure	Selected batch size
Measurement	50 per-operation batch means	50 per-operation batch means	Mean, median, sample SD, min, max, P95, CI
Independent check	Representative native executable	Representative native executable	Eight program return codes

Correctness gates reduce the risk of timing an error-return path or a benchmark that silently omits verification. They do not establish resistance to malformed-ciphertext timing attacks, fault injection, cache attacks, electromagnetic leakage, or other side channels. They also do not prove that buffers are erased or that random-number generation is suitable for a production key lifecycle. These requirements remain implementation-assurance tasks.

3.3. Adaptive-Batch Timing

Sub-millisecond native operations are difficult to measure as isolated Python calls because orchestration and timer granularity may become a visible fraction of the sample. The benchmark therefore calibrates a batch size B by doubling it until the batch approaches a 20 ms target. If $T_j(B)$ is the wall-clock time of batch j, the per-operation observation is

$$x_j = \frac{T_j(B)}{B}, j = 1, \ldots, N. \tag{7}$$

The number of timed batch observations per operation within the primary session was $N = 50$. Ten untimed warm-up calls preceded calibration. The arithmetic mean is

$$\overline{x} = \frac{1}{N}\sum_{j=1}^{N} x_j. \tag{8}$$

The sample standard deviation is

$$s = \sqrt{\frac{1}{N-1}\sum_{j=1}^{N}(x_j - \overline{x})^2}. \tag{9}$$

The reported within-session 95% confidence interval uses the normal-approximation half-width

$$h_{0.95} = z_{0.975}\frac{s}{\sqrt{N}} \approx 1.96\frac{s}{\sqrt{N}}, \tag{10}$$

and is written as

$$CI_{0.95} = [\overline{x} - h_{0.95}, \overline{x} + h_{0.95}]. \tag{11}$$

The interval is conditional on the recorded batches and the normal-approximation assumptions. It does not correct autocorrelation or include between-session variability, and its nominal coverage may therefore be inaccurate. Median, minimum, maximum, and an interpolated sample P95 are retained in the workbook. Throughput is a derived quantity,

$$Throughput = \frac{1000}{\bar{x}} \text{operations/s}, \tag{12}$$

where $\bar{x}$ is expressed in milliseconds. Throughput is not separately measured under concurrency and must not be interpreted as requests per second for a network service.

All timing comparisons are descriptive single-session snapshots for the tested configurations. The 50 observations per operation are repeated timed batches within a session, not 50 independent sessions. Library order was not randomized or interleaved across independent days, and batches may be autocorrelated. The reported normal-approximation intervals concern within-session variation only and do not establish reproducibility across sessions. Ratios cannot isolate a library or backend effect from temporal system drift. Section 6.2 uses the earlier 30-observation run only as a sensitivity diagnostic, not as a replicated performance study.

3.4. Evidence Provenance and Comparison Rules

The experimental data package separates local measured rows, derived statistics, explicit estimates, and theoretical statements. Four rules govern comparison. First, liboqs and PQMagic are compared only for the same standardized algorithm and parameter set, with PQMagic using SHAKE. Second, SM3 and SHAKE are compared only within PQMagic for the same algorithm-operation tuple. Third, Aigis modes are described with their own names, sizes, and source-paper security context. Fourth, Abbasi et al. values remain external measured evidence and are not pooled with local observations.

Artifact sizes are obtained from the library interfaces and algorithm specifications. A KEM communication artifact is summarized as $S_{pk} + S_{ct}$, and a signature artifact as $S_{pk} + S_{sig}$. These are derived payload quantities, not complete TLS records or X.509 objects. The limited certificate illustration uses the deliberately simple additive model

$$S_{cert}^{est} = S_{pk} + S_{sig} + 300B. \tag{13}$$

The 300-byte term is a round illustrative allowance for certificate fields other than the PQC public key and signature. It was not calibrated from a certificate corpus or generated profile. Sensitivity is linear: replacing 300 B with an allowance A changes every modeled PQC total by A − 300 B and does not change their ordering because the same allowance is applied to each row. The term should therefore be replaced by a measured profile-specific value before deployment use. In the later certificate-size results, RSA rows are locally generated DER certificates and are labeled measured; PQC rows remain estimated. Zero placeholders for PQC generation time are excluded.

3.5. Reproducibility Package

The supplementary archive Supplementary_Materials_R7.zip (SHA-256: b498fe9827e77b145e174bf7d375926e76a499c11b17c6775fd6051e9bd2b26f) contains the publication workbooks, case records, five complete liboqs ACVP projections, current and baseline binaries, the native harness, build records including six Ninja rule files and the recovered static liboqs archive, and source data and code for all fourteen figures. The 10,146 referenced binary payloads are included and match every recorded hash and length; the 2005 JSONL rows include 13 not-applicable boundaries. Project-relative paths are retained under recovered_project/. The v13 and earlier-run individual timing batches were not retained;

the 4800 v14 observations belong to a separate session. Upstream requests/responses and the downloaded ACVP release archive were not retained; full sources and toolchain installations remain external rebuild prerequisites. MISSING_MATERIALS.md and the read-only verification script distinguish file auditability from independently rerun experiments. No native binary was executed during this revision.

3.6. Native Conformance and Interoperability Architecture

The native harness treats each parameter set as an immutable descriptor containing the algorithm kind and exact public-key, secret-key, ciphertext or signature, and shared-secret lengths. liboqs is linked from commit '97f6b86b1b6d109cfd43cf276ae39c2e776aed80'; PQMagic DLLs are loaded from commit '9613aa3c2eb5b242b839efe99d852f5addd73d77'. Before a raw function pointer is called, the descriptor gates every buffer length. The PQMagic adapter resolves 33 public and deterministic internal symbols and does not fall back from a missing deterministic symbol to a random public entry point. The two projects are treated as separate library distributions; source-code independence is not inferred from repository identity.

The available build scripts and manifests record the following configuration parameters. PQMagic used unversioned Ninja, LLVM Clang 20.1.2, and CMake 4.4.0; CMAKE_BUILD_TYPE = Release, CMAKE_WINDOWS_EXPORT_ALL_SYMBOLS = ON, ENABLE_ML_KEM/ML_DSA/AIGIS_ENC/AIGIS_SIG = ON, ENABLE_KYBER/DILITHIUM/SLH_DSA/SPHINCS_A = OFF, ENABLE_BENCH = ON, USE_SHAKE = ON and USE_SM3 = OFF for the SHAKE build (the SM3 build swaps these last two flags). liboqs used the same Ninja/Clang toolchain, CMAKE_BUILD_TYPE = Release, BUILD_SHARED_LIBS = ON, OQS_BUILD_ONLY_LIB = ON, OQS_USE_OPENSSL = OFF and the recorded minimal algorithm set; the vector helper build used BUILD_SHARED_LIBS = OFF, OQS_BUILD_ONLY_LIB = OFF and MINGW = ON. The native C harness used CMake release configuration with CMAKE_PREFIX_PATH set to the liboqs install tree and compiled the adapter sources into one executable linked through the imported static target OQS::oqs_static. The supplementary scripts retain configuration command arrays, tool paths, commits, binary hashes, and target architecture x86_64-w64-mingw32. The recovered harness CMakeLists.txt and Ninja graph specify C11, -O3-DNDEBUG-Wall-Wextra-Wpedantic-Werror, and linkage to the static liboqs archive. The library graphs record per-target flags, including PQMagic native-tuning options and liboqs shared-link settings; Supplementary BUILD_CONFIGURATION_DETAILS.md indexes them. CMake caches, both liboqs compile databases, and available logs are included. All six CMakeFiles/rules.ninja files and the static archive named by the link graph are now included and match the supplied recovery checksums. Some compile databases and the harness build transcript were not retained; full source and toolchain installations remain external prerequisites. The saved settings are auditable, but a complete rebuild and binary equivalence have not been independently verified.

The native C harness executes a command-dispatched contract: immutable algorithm descriptors provide exact buffer lengths; adapters resolve library symbols and reject missing deterministic entry points; file inputs are parsed as hexadecimal or binary artifacts; each operation checks return codes, output lengths, and expected hashes; KEM decapsulation compares shared-secret hashes and records implicit-rejection outcomes; ML-DSA verification records the library Boolean result; and every case is emitted as a JSONL record with algorithm, direction, case identifier, expected outcome, actual outcome, binary hashes, and error classification. Producer and consumer runs are separate processes for replay to avoid loading two same-named PQMagic DLLs. The full C sources and adapter headers are included in the supplementary code directory.

The evidence layers in Table 5 are dependency ordered. Crucially, self-consistency does not establish interoperability. An implementation can generate and consume a mutually consistent but non-standard encoding; only exact official vectors test the specified bytes, and only a cross-implementation consumer tests replaceability at the artifact boundary. See Figure 1.

Table 5. Native evidence layers, required results, and primary failure classifications.

Layer	Executed Evidence	Required Result	Failure Classification
Build contract	Pinned commits, binary hashes, 33 PQMagic symbols	Exact descriptor and symbol closure	Infrastructure failure
Official vectors	FIPS 203/204 ACVP projections	Exact bytes or expected verification result	VECTOR_MISMATCH
Positive interoperability	Two KEM and two signature directions	Shared-secret equality or successful verification	INTEROP_FAILURE
Negative cases	Deterministic one-bit mutations	Rejection; stable ML-KEM implicit-rejection secret	INTEROP_FAILURE
Supporting same-commit replay	Archived/clean builds of one commit; producer–consumer subprocesses	Recorded artifact remains consumable for the tested build pair	BUILD_REPLAY_FAILURE

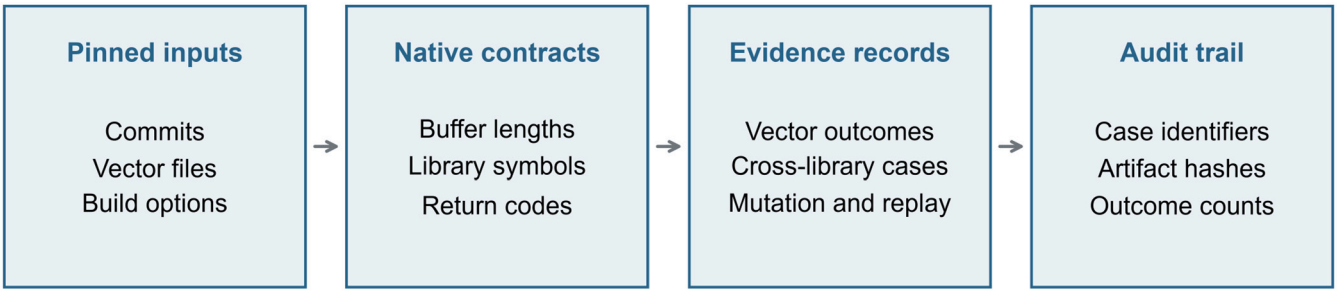


Figure 1. Conformance and raw-artifact interoperability workflow. Case records preserve outcomes and artifact provenance; timing and configuration summaries provide supporting context.

3.7. Vector Selection, Bidirectional Cases, and Mutation Policy

Five ACVP projections present in the pinned liboqs checkout were used, following the NIST ACVP documentation [33]. The files were 'ML-KEM-keyGen-FIPS203', 'ML-KEM-encapDecap-FIPS203', 'ML-DSA-keyGen-FIPS204', 'ML-DSA-sigGen-FIPS204', and 'ML-DSA-sigVer-FIPS204', each with 'internalProjection.json'. Their checkout-relative paths, SHA-256 values, and the liboqs commit '97f6b86b1b6d109cfd43cf276ae39c2e776aed80' are listed in Supplementary Table S1 and 'manifests/vector_manifest.json'. The selected set covers FIPS 203 key generation, encapsulation, and decapsulation, and FIPS 204 key generation, pure external-interface signing, and verification. ML-DSA signing records require 'externalMu = false', deterministic signing, and include empty and non-empty contexts where present. The archive retains record-level projection name, test-group and test-case identifiers, expected and actual hashes, and outcomes. All five complete projection files and their manifest hashes are retained. The recovered fetch_values.sh identifies ACVP-Server v1.1.0.40 and directly extracts its gen-val/json-files projections; it does not generate them from requests/responses. prepare_v25_interop_inputs.py freezes hashes, while run_v25_acvp.py selects groups and decodes inputs and expected outputs

for the native harness. The exact predicates and source paths are listed in Supplementary ACVP_PROVENANCE.csv and .md. The original upstream request/response files and downloaded release archive were not retained, so their historical bytes were not independently checked. Local request.json files describe harness calls, not upstream ACVP requests. Unselected interfaces, prehash, external-mu, and future vector variants remain outside the claim.

For each of six parameter sets and each applicable cross-library direction, 100 cases are derived with SHAKE256 domain separation. KEM cases exchange public keys and ciphertexts and compare both shared-secret hashes. Signature cases exchange public keys, messages, contexts, and detached signatures and require the other library to verify. Negative cases flip mask '0x01' at offsets 0, $\lfloor L/2 \rfloor$, and $L - 1$. Corrupted ML-KEM ciphertexts are decapsulated twice; the rejection secrets must be stable and unequal to the valid secret. Replay runs the producer and consumer in separate processes so only one same-named PQMagic DLL is loaded at a time.

In the result vocabulary, 'EXPECTED_REJECT' has primitive-specific meaning. ML-DSA verification returns failure for mutated inputs, whereas ML-KEM decapsulation returns a deterministic implicit-rejection secret that must differ from the valid shared secret. This deliberately sparse three-offset design checks stable behavior at the tested library boundary. It is not malformed-input testing, structural parsing analysis, or fuzzing, and it does not cover invalid lengths, truncation, multi-bit corruption, or arbitrary noncanonical encodings.

4. Results

4.1. Supporting Broad KEM and Signature Baselines

Sections 4.1–4.8 provide supporting performance and engineering context; the primary conformance, interoperability, and negative-case results appear in Sections 4.9–4.11. Table 6 is a descriptive cross-family snapshot. Its second and third operation columns are primitive-specific and do not imply identical KEM semantics. They denote encapsulation and decapsulation for KEMs, peer-key agreement paths for X25519/ECDH, and configured public/private operations for RSA. These rows are not evidence for substituting unlike primitives or for stable cross-session performance rankings.

Table 6. Local KEM and classical key-exchange baseline; measured, 50 observations. Means ± within-session 95% normal-approximation half-widths; intervals exclude between-session variation.

Primitive/ Algorithm	KeyGen Mean $\pm h_{0.95}$ (ms)	Public/Peer Operation Mean (ms)	Private Operation Mean (ms)	Public Key (B)	Peer Artifact (B)
X25519	0.2991 ± 0.4549	0.0308 ± 0.0024	0.0292 ± 0.0002	32	32
ECDH-P256	0.0419 ± 0.0115	0.0588 ± 0.0019	0.0580 ± 0.0022	91	91
RSA-2048	42.1939 ± 6.2774	0.0782 ± 0.0131	0.6681 ± 0.0217	294	256
RSA-3072	144.6304 ± 25.3711	0.0957 ± 0.0037	1.4027 ± 0.0210	422	384
RSA-4096	447.0856 ± 77.7060	0.1321 ± 0.0095	2.5197 ± 0.0294	550	512
ML-KEM-512	0.2138 ± 0.0124	0.2069 ± 0.0044	0.0292 ± 0.0004	800	768
ML-KEM-768	0.2161 ± 0.0051	0.2159 ± 0.0036	0.0413 ± 0.0006	1184	1088
ML-KEM-1024	0.2368 ± 0.0063	0.2413 ± 0.0084	0.0581 ± 0.0005	1568	1568
Classic-McEliece-348864	194.3801 ± 43.4381	0.5462 ± 0.0999	13.9493 ± 0.1800	261,120	96
FrodoKEM-640-AES	0.5002 ± 0.0385	0.6093 ± 0.0318	0.3526 ± 0.0027	9616	9720

Figure 2 uses logarithmic axes because RSA and Classic McEliece would otherwise compress the sub-millisecond observations. The dot plot is not a single score. The left panel

emphasizes key-generation cost, the middle panel distinguishes public/peer and private operations, and the right panel displays public and peer artifacts. The different rankings across panels illustrate why deployment selection needs an operation and transport model.

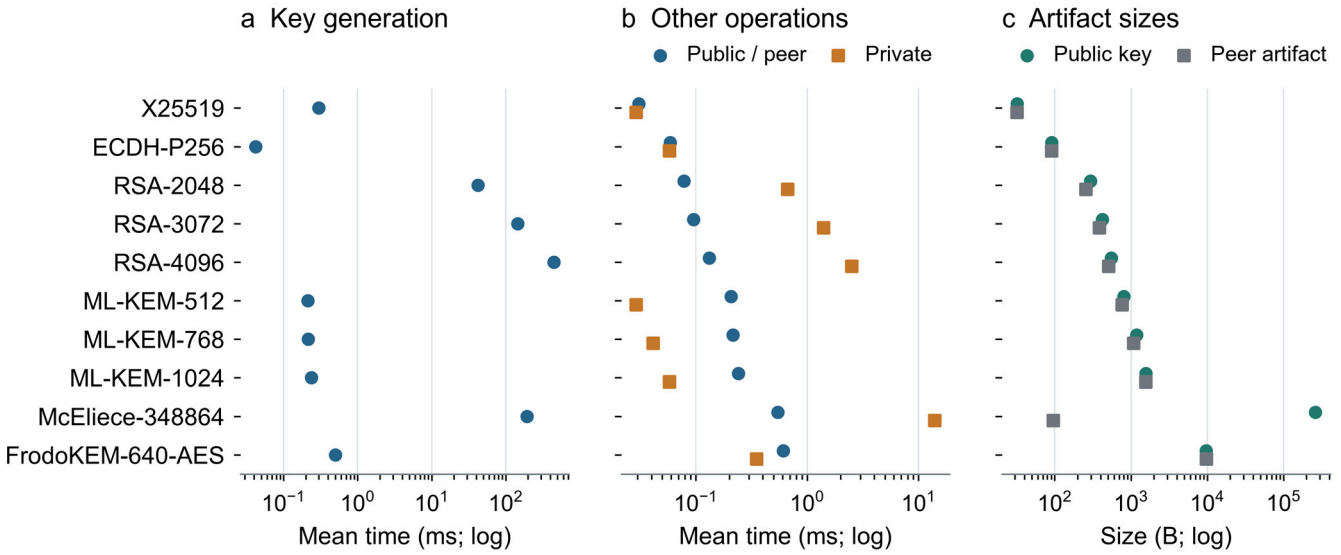


Figure 2. Supporting KEM and classical baseline: (**a**) key generation, (**b**) primitive-specific public/peer and private operations, and (**c**) artifact sizes. Points show recorded means on logarithmic axes. All ten algorithms are retained. Within-session intervals remain in Table 6; they are omitted here because some unadjusted normal intervals extend below zero. These are descriptive measurements, not stable rankings.

Table 7 reports the signature baseline under the configured message and signature modes. ECDSA P-256 and Ed25519 remain compact. ML-DSA signatures are larger but provide standardized lattice paths. Falcon is compact relative to ML-DSA but has a more demanding implementation profile. RSA exhibits a strong private/public asymmetry, and key generation dominates its local cost. These measurements support scenario-specific comparisons rather than a global signature ranking. Figure 3 visualizes the corresponding operation means and artifact sizes.

Table 7. Local signature baseline; measured, 50 observations. Means ± within-session 95% normal-approximation half-widths; intervals exclude between-session variation.

Algorithm	KeyGen Mean ± $h_{0.95}$ (ms)	Sign Mean ± $h_{0.95}$ (ms)	Verify Mean ± $h_{0.95}$ (ms)	Public Key (B)	Signature (B)
ECDSA-P256	0.0174 ± 0.0018	0.0394 ± 0.0140	0.0666 ± 0.0011	91	71
Ed25519	0.0334 ± 0.0010	0.0334 ± 0.0013	0.0936 ± 0.0020	32	64
RSA-2048	38.5568 ± 6.2677	0.6528 ± 0.0126	0.0437 ± 0.0042	294	256
RSA-3072	128.1143 ± 18.8101	1.4169 ± 0.0114	0.0632 ± 0.0007	422	384
RSA-4096	451.9183 ± 86.2340	2.5784 ± 0.0340	0.0990 ± 0.0031	550	512
ML-DSA-44	0.2783 ± 0.0119	0.4685 ± 0.0496	0.0773 ± 0.0020	1312	2420
ML-DSA-65	0.3345 ± 0.0135	0.5804 ± 0.0705	0.1220 ± 0.0050	1952	3309
ML-DSA-87	0.3964 ± 0.0126	0.6984 ± 0.0781	0.1910 ± 0.0051	2592	4627
Falcon-512	4.9984 ± 0.4213	0.6247 ± 0.0234	0.0497 ± 0.0043	897	656
Falcon-1024	14.9868 ± 1.4783	0.8937 ± 0.0686	0.0886 ± 0.0086	1793	1270

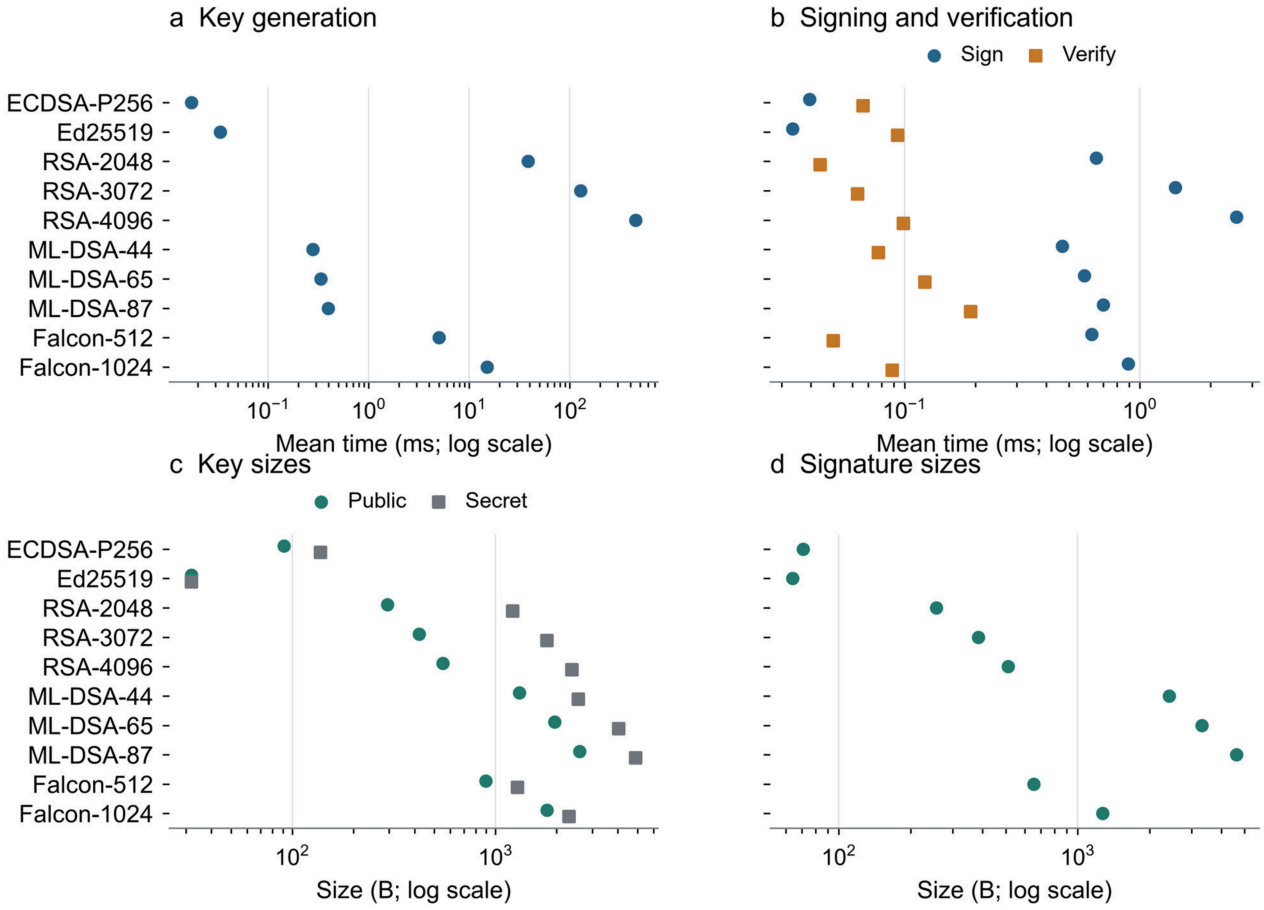


Figure 3. Supporting signature baseline: (**a**) key generation, (**b**) signing and verification, (**c**) key sizes, and (**d**) signature sizes. Points show recorded means on logarithmic axes; all ten algorithms are retained. Within-session intervals remain in Table 7 and are not plotted because some normal-approximation limits fall outside the positive log domain.

4.2. ML-KEM Comparison Between Liboqs and PQMagic

Table 8 reports the aligned ML-KEM comparison dataset. PQMagic/liboqs mean-time ratios range from 0.879 to 1.374. PQMagic was faster for ML-KEM-1024 encapsulation in this session, with a ratio of 0.879, but slower for the other encapsulation parameter sets and all three decapsulation paths. The largest ratio occurred for ML-KEM-768 key generation (1.374). These descriptive single-session results do not support a library-wide multiplier. Figure 4 visualizes the three aligned ML-KEM operations.

Table 8. Aligned ML-KEM operation comparison; measured means and derived ratios. Time entries are means ± within-session 95% normal-approximation half-widths; ratios describe the recorded session.

Algorithm	Operation	Liboqs (ms)	PQMagic-SHAKE (ms)	PQMagic/Liboqs
ML-KEM-1024	Decaps	0.0599 ± 0.0017	0.0727 ± 0.0018	1.213
ML-KEM-1024	Encaps	0.2804 ± 0.0076	0.2465 ± 0.0112	0.879
ML-KEM-1024	KeyGen	0.3101 ± 0.0097	0.3507 ± 0.0161	1.131
ML-KEM-512	Decaps	0.0279 ± 0.0007	0.0364 ± 0.0010	1.306
ML-KEM-512	Encaps	0.2118 ± 0.0047	0.2902 ± 0.0216	1.370
ML-KEM-512	KeyGen	0.2370 ± 0.0082	0.2548 ± 0.0126	1.075
ML-KEM-768	Decaps	0.0417 ± 0.0013	0.0515 ± 0.0009	1.235
ML-KEM-768	Encaps	0.2687 ± 0.0073	0.2917 ± 0.0139	1.086
ML-KEM-768	KeyGen	0.2867 ± 0.0108	0.3938 ± 0.0198	1.374

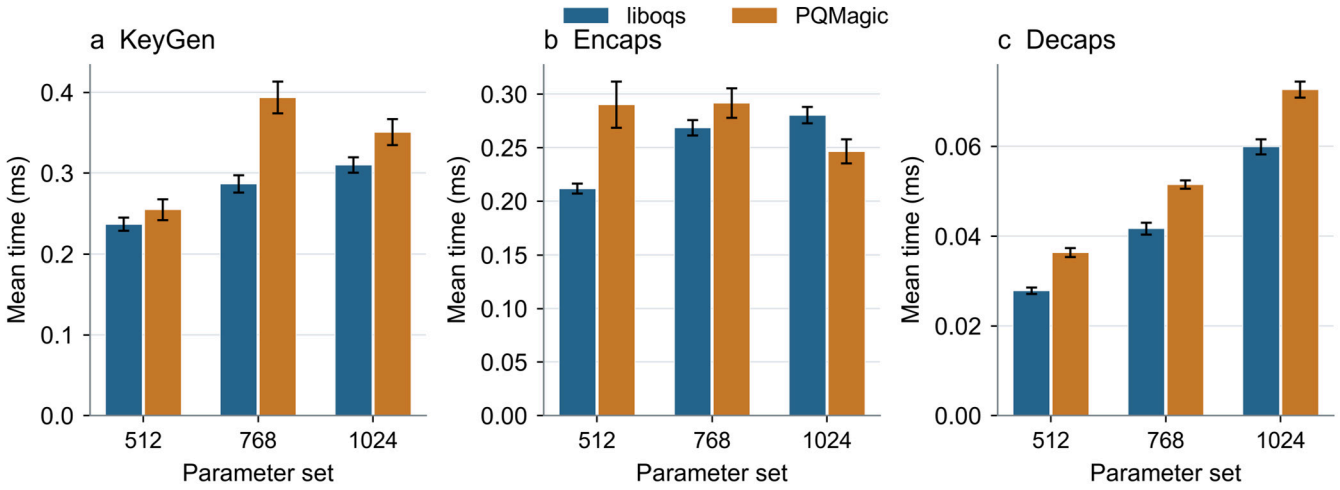


Figure 4. Same-host ML-KEM comparison: (**a**) KeyGen, (**b**) Encaps, and (**c**) Decaps. Bars show means; whiskers show within-session 95% normal-approximation intervals from 50 timed batches per operation. Intervals exclude between-session variability; comparisons are descriptive.

The differences are large enough to matter in a microbenchmark but small compared with RSA key generation and with many protocol or network delays. For example, an implementation difference of tenths of a millisecond may be important in a high-rate termination service, while the public-key and ciphertext sizes may dominate on a constrained link. The result also resists an attractive but unsupported explanation. Although the host advertises vector instructions, this study did not trace executed instructions. It would therefore be incorrect to attribute a particular ratio to AVX2, AVX-512, or Keccak assembly without path-level evidence.

4.3. ML-DSA Comparison Between Liboqs and PQMagic

The aligned ML-DSA dataset showed a larger directional difference for signing. Table 9 reports PQMagic-SHAKE/liboqs signing ratios of 0.188, 0.236, and 0.412. Verification ratios were 0.795, 0.862, and 0.877. Key generation was mixed: PQMagic was faster for ML-DSA-44 and ML-DSA-87 and approximately equal for ML-DSA-65. These ratios describe the recorded session and do not isolate a causal library effect from temporal system variation. Figure 5 visualizes the three aligned ML-DSA operations.

Table 9. Aligned ML-DSA operation comparison; measured means and derived ratios. Time entries are means ± within-session 95% normal-approximation half-widths; ratios describe the recorded session.

Algorithm	Operation	Liboqs (ms)	PQMagic-SHAKE (ms)	PQMagic/Liboqs
ML-DSA-44	KeyGen	0.3306 ± 0.0115	0.2593 ± 0.0128	0.784
ML-DSA-44	Sign	0.5342 ± 0.0109	0.1004 ± 0.0020	0.188
ML-DSA-44	Verify	0.0762 ± 0.0011	0.0606 ± 0.0011	0.795
ML-DSA-65	KeyGen	0.3130 ± 0.0109	0.3158 ± 0.0146	1.009
ML-DSA-65	Sign	0.6857 ± 0.0248	0.1616 ± 0.0039	0.236
ML-DSA-65	Verify	0.1197 ± 0.0026	0.1032 ± 0.0025	0.862
ML-DSA-87	KeyGen	0.4340 ± 0.0134	0.3416 ± 0.0071	0.787
ML-DSA-87	Sign	0.8039 ± 0.0306	0.3315 ± 0.0054	0.412
ML-DSA-87	Verify	0.1925 ± 0.0046	0.1689 ± 0.0039	0.877

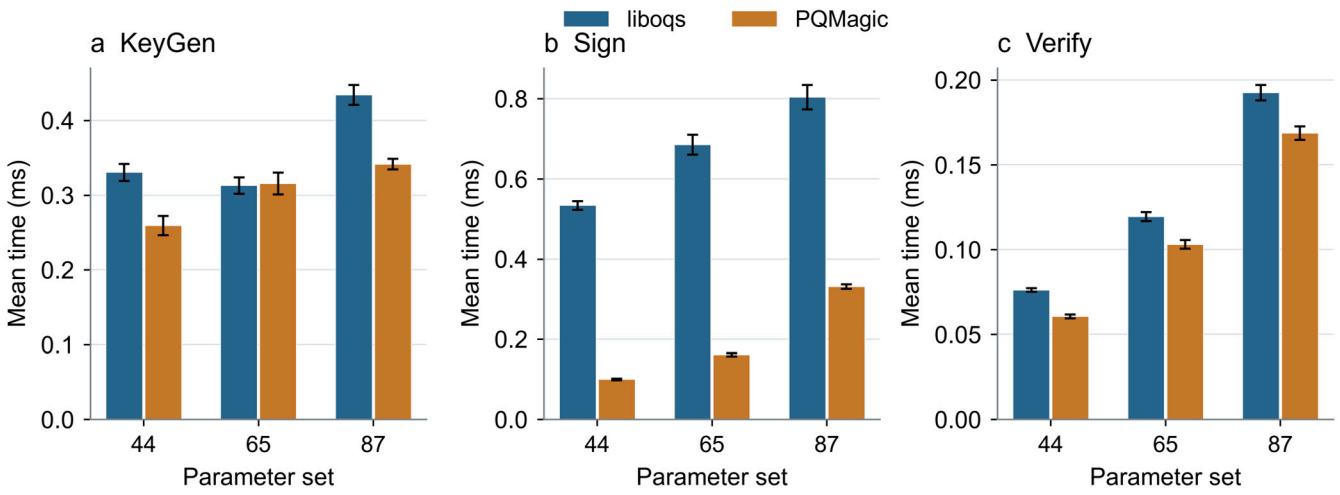


Figure 5. Same-host ML-DSA comparison: (**a**) KeyGen, (**b**) Sign, and (**c**) Verify. Bars show means; whiskers show within-session 95% normal-approximation intervals from 50 timed batches per operation. They do not establish a stable library effect across sessions.

The lower recorded signing means may be operationally relevant for services that execute private operations frequently. They are not sufficient for implementation selection. Production acceptance also depends on random-number generation, protected key storage, constant-time behavior, update policy, interface semantics, and interoperability. The bounded result is that PQMagic-SHAKE recorded lower mean ML-DSA signing times under the tested build, host, message, mode, and timing protocol.

4.4. Supporting Aigis-Enc and Aigis-Sig Measurements

Table 10 combines the SHAKE measurements for all Aigis modes. Aigis-Enc mode 2, a useful 128-bit-target reference from the source construction, records means of 0.2178 ± 0.0082 ms for KeyGen, 0.3109 ± 0.0112 ms for Encaps, and 0.0562 ± 0.0017 ms for Decaps. Its 896-byte public key and 992-byte ciphertext are smaller than ML-KEM-768's corresponding artifacts, but the schemes should not be substituted without protocol identifiers and policy. Figure 6 visualizes the Aigis-Enc operation means and artifact sizes.

Table 10. Aigis SHAKE operation times and artifact sizes; measured, 50 observations. Time entries are means ± within-session 95% normal-approximation half-widths.

Algorithm	Operation Means ± $h_{0.95}$ (ms)	Public Key (B)	Ciphertext/Signature (B)	Three-Operation Sum (ms)
Aigis-Enc-1	KeyGen 0.2179 ± 0.0089; Encaps 0.2549 ± 0.0107; Decaps 0.0410 ± 0.0010	672	736	0.5138
Aigis-Enc-2	KeyGen 0.2178 ± 0.0082; Encaps 0.3109 ± 0.0112; Decaps 0.0562 ± 0.0017	896	992	0.5850
Aigis-Enc-3	KeyGen 0.2017 ± 0.0065; Encaps 0.2994 ± 0.0110; Decaps 0.0555 ± 0.0014	992	1056	0.5567
Aigis-Enc-4	KeyGen 0.2652 ± 0.0113; Encaps 0.3055 ± 0.0092; Decaps 0.0844 ± 0.0026	1440	1568	0.6552
Aigis-Sig-1	KeyGen 0.2304 ± 0.0104; Sign 0.4046 ± 0.0045; Verify 0.0558 ± 0.0014	1056	1852	0.6908

Table 10. *Cont.*

Algorithm	Operation Means ± $h_{0.95}$ (ms)	Public Key (B)	Ciphertext/Signature (B)	Three-Operation Sum (ms)
Aigis-Sig-2	KeyGen 0.2823 ± 0.0064; Sign 0.2967 ± 0.0059; Verify 0.0794 ± 0.0013	1312	2445	0.6585
Aigis-Sig-3	KeyGen 0.2903 ± 0.0094; Sign 0.4258 ± 0.0113; Verify 0.1087 ± 0.0033	1568	3046	0.8248

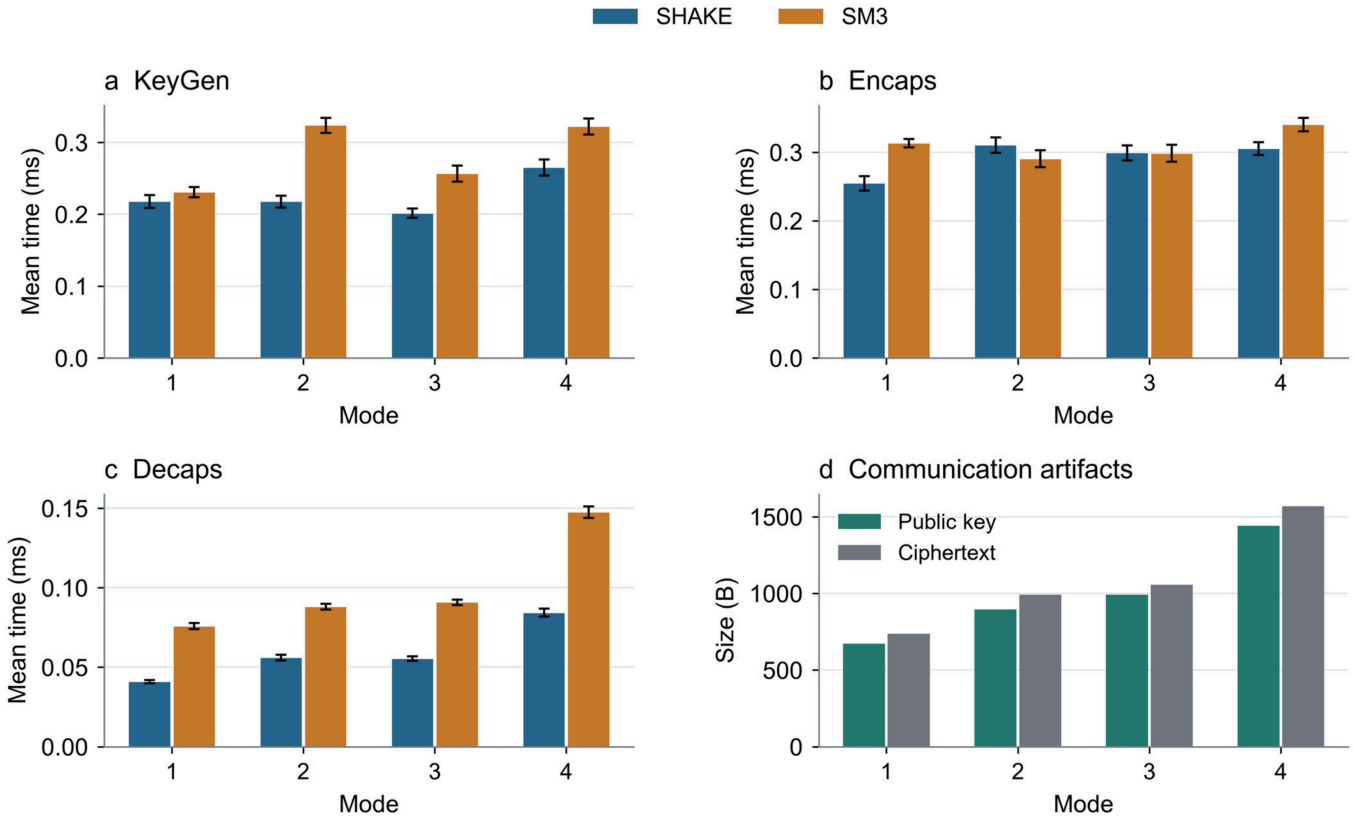


Figure 6. Supporting Aigis-Enc observations: (**a**–**c**) operation means for SHAKE and SM3, with within-session 95% normal-approximation intervals from 50 batches, and (**d**) public-key and ciphertext sizes. Backend comparisons are descriptive; artifact sizes are identical across the two tested backends.

Aigis-Sig mode 2 has a 1312-byte public key and 2445-byte signature. Its SHAKE signing mean is 0.2967 ms. Mode numbering is not a universal security scale shared with NIST categories. The source paper provides parameter-specific estimates and reductions [27]; this experiment contributes implementation observations. A deployment that selects Aigis for domestic supply-chain or algorithm-diversity reasons should preserve that distinction in documentation and key metadata. Figure 7 visualizes the corresponding Aigis-Sig observations.

4.5. Supporting SHAKE-to-SM3 Backend Comparison

Table 11 summarizes the derived ratio $r_{SM3/SHAKE} = \bar{x}_{SM3}/\bar{x}_{SHAKE}$. A value above one means the SM3 build was slower for that tuple. The observed ratios span 0.741–7.481. Table 11 pools operations only for compact description; Figure 8 retains the operation-level comparison.

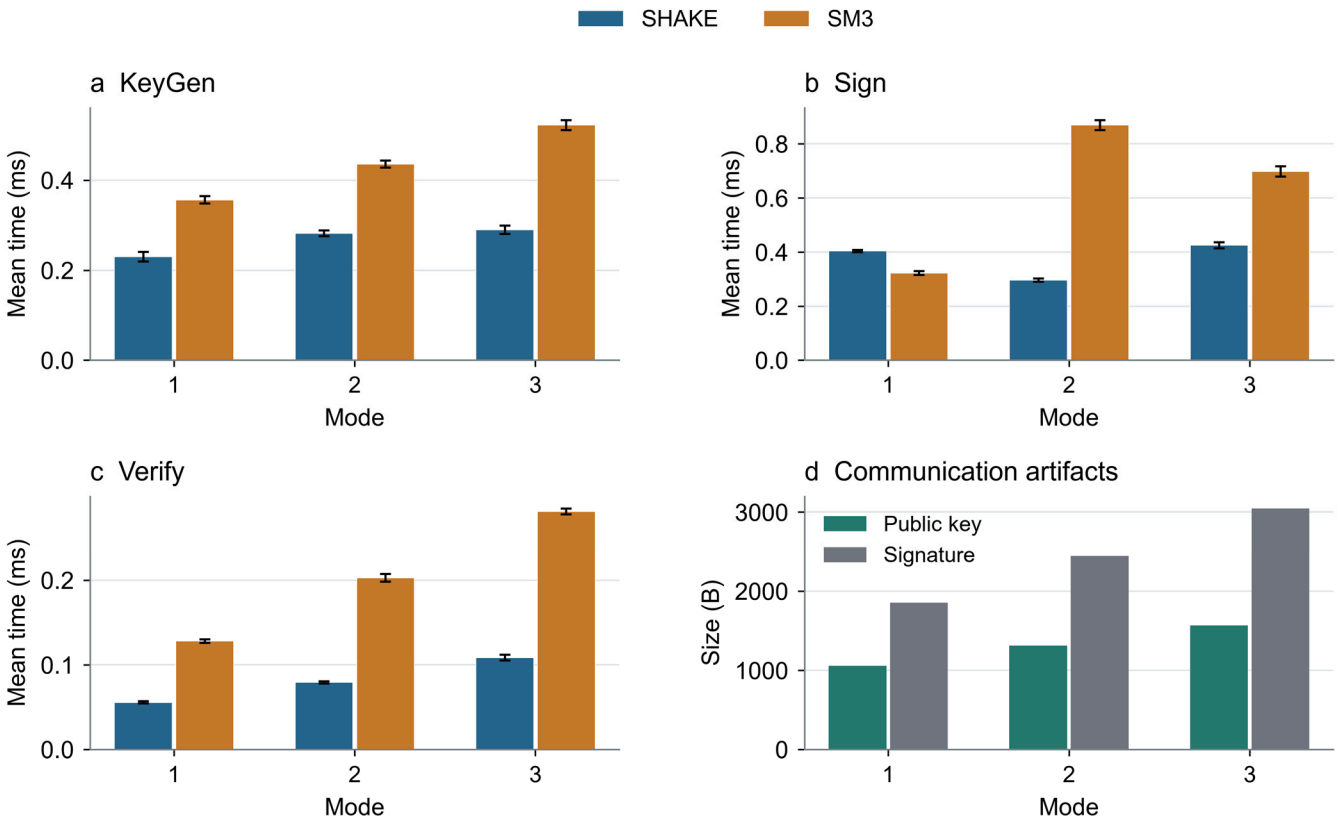


Figure 7. Supporting Aigis-Sig observations: (**a**–**c**) operation means for SHAKE and SM3, with within-session 95% normal-approximation intervals from 50 batches, and (**d**) public-key and signature sizes. Backend comparisons are descriptive and do not extend the standardized interoperability claim.

Table 11. Family-level SM3/SHAKE mean-time ratios; derived descriptive summaries of the recorded session, not workload scores.

Family	Mean Ratio	Median Ratio	Minimum	Maximum
Aigis-Enc	1.344	1.251	0.935	1.853
Aigis-Sig	1.968	1.800	0.799	2.931
ML-DSA	2.881	2.394	1.377	7.481
ML-KEM	1.198	1.022	0.741	1.693

Figure 8 identifies which operations had higher SM3 mean times across every tested parameter set within this recorded session. ML-KEM decapsulation ratios were 1.462–1.693. For ML-DSA, all key-generation, signing, and verification ratios exceeded one, with respective ranges of 1.377–2.102, 1.661–7.481, and 2.394–2.671. Aigis-Enc key generation and decapsulation likewise exceeded one in all four modes, with ranges of 1.060–1.487 and 1.570–1.853. Aigis-Sig key generation and verification exceeded one in all three modes, with ranges of 1.545–1.800 and 2.301–2.589. These patterns identify ML-DSA signing, signature verification, and KEM decapsulation as priorities for workload-specific backend retesting. They describe the measured configurations, not intrinsic SM3 costs or consistency over time.

No operation was observed to match or outperform SHAKE across every parameter set of an algorithm family. Lower SM3 means occurred for ML-KEM-768/1024 key generation (0.741/0.846), ML-KEM-512 encapsulation (0.895), Aigis-Enc-2 encapsulation (0.935), and Aigis-Sig-1 signing (0.799). Aigis-Enc-3 encapsulation was near unity at 0.998, an observed difference of about 0.2%, not demonstrated statistical equivalence. The remaining ML-KEM key-generation and encapsulation ratios exceeded one, as did Aigis-Sig-2/3 signing and

Aigis-Enc-1/4 encapsulation. These tuple-specific observations support targeted retesting, not a claim of consistent SM3 superiority across sessions. The comparison does not establish that replacing SHAKE with SM3 preserves the standardized ML-KEM/ML-DSA profile.

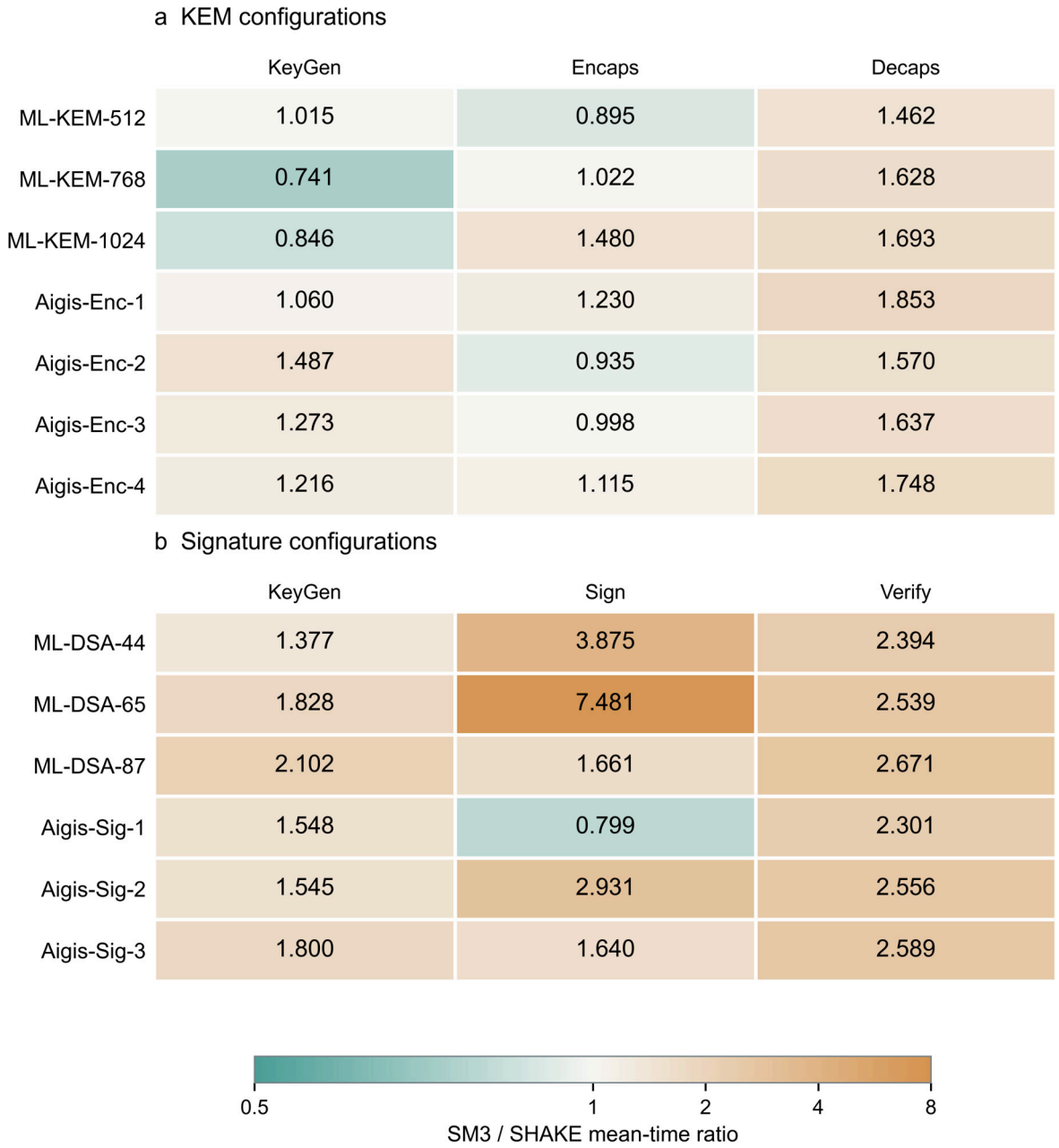


Figure 8. All 39 recorded SM3/SHAKE mean-time ratios, grouped into (**a**) KEM and (**b**) signature configurations. Values above one indicate higher SM3 means. The diverging color scale is centered at one and transforms ratios logarithmically; exact ratios are printed to three decimals. Cross-parameter patterns do not establish temporal consistency or equivalence.

4.6. Certificate Payload Provenance

Table 12 separates locally generated RSA DER certificate sizes from illustrative PQC payload estimates. For the PQC rows, the model adds the public-key size, signature size, and a common 300-byte allowance for all remaining certificate structure. That round allowance is not calibrated from a generated X.509 profile. If a deployment replaces it with A bytes, each PQC total changes by A − 300 B; the relative ordering of the modeled PQC rows is unchanged, but every absolute total changes directly. The model does not

represent a generated certificate, chain, protocol trace, fragmentation result, or compression measurement. Figure 9 visualizes the measured and illustrative values on a common byte scale.

Table 12. Measured RSA DER certificates and estimated PQC payloads.

Algorithm	Size (B)	Evidence	Method
RSA-2048	891	Measured	DER certificate generated locally
RSA-3072	1147	Measured	DER certificate generated locally
RSA-4096	1403	Measured	DER certificate generated locally
ML-DSA-44	4032	Estimated	public key + signature + 300 B fixed structural allowance
ML-DSA-65	5561	Estimated	public key + signature + 300 B fixed structural allowance
ML-DSA-87	7519	Estimated	public key + signature + 300 B fixed structural allowance
Falcon-512	1848	Estimated	public key + signature + 300 B fixed structural allowance
Falcon-1024	3360	Estimated	public key + signature + 300 B fixed structural allowance

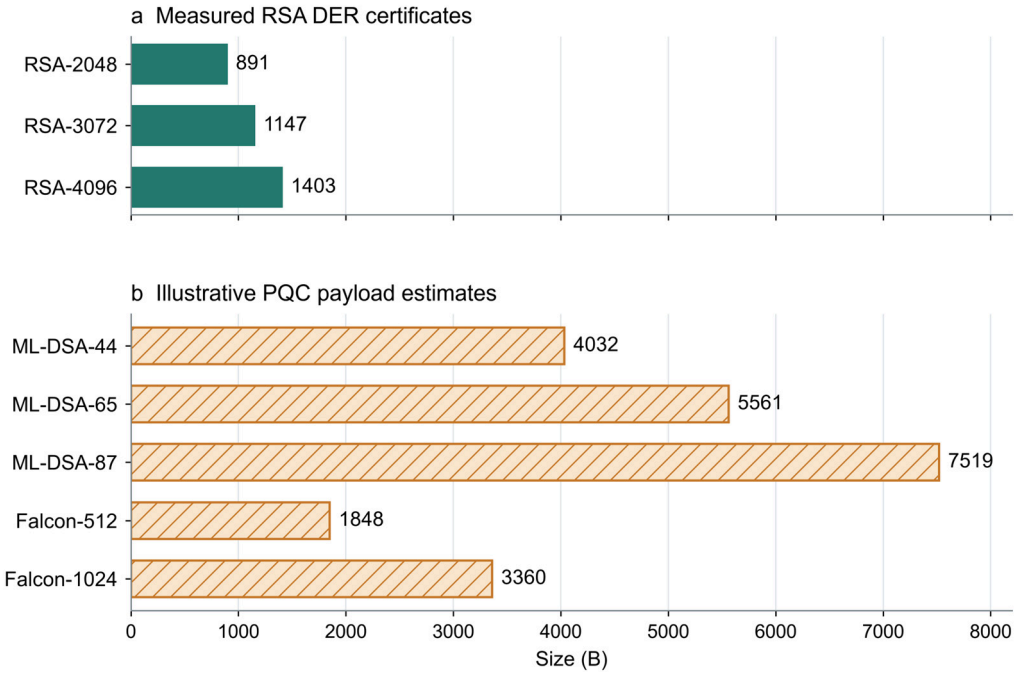


Figure 9. (**a**) Locally generated RSA DER certificate sizes and (**b**) illustrative PQC payload estimates, shown on the same byte scale with distinct fill patterns. Hatched estimates add a public key, signature and uncalibrated 300 B allowance; they are not generated X.509 certificates or handshake measurements.

The operational viewpoint is straightforward. ML-DSA's larger signatures can be acceptable for software updates or authentication events that are infrequent relative to application data. They can be more consequential in certificate chains, constrained MTUs, or repeated handshakes. Falcon's compact signature may reduce transport cost, but that benefit must be balanced against implementation assurance. Artifact size is therefore a scenario variable rather than a reason to declare a universal winner.

4.7. Quantitative Comparison with Abbasi et al.

Table 13 compares representative local liboqs means with the Laptop E2 means reported by Abbasi et al. [7]. The complete 18-row comparison is available in the supplementary dataset. Their experiment used 1000 trials and a different laptop, implementation environment, and timing protocol. Local/external ratios are thus descriptive rather than causal. Figure 10 visualizes all 18 operation-level ratios.

Table 13. Descriptive comparison with Abbasi et al.; unmatched measured platforms.

Algorithm	Operation	This Work: Local Liboqs (ms)	Abbasi Laptop E2 (ms)	Local/External	External Source
ML-KEM-768	KeyGen	0.2867 ± 0.0108	0.280	1.024	Table 4
ML-KEM-768	Encaps	0.2687 ± 0.0073	0.220	1.221	Table 4
ML-KEM-768	Decaps	0.0417 ± 0.0013	0.250	0.167	Table 4
ML-DSA-65	KeyGen	0.3130 ± 0.0109	0.360	0.870	Table 4
ML-DSA-65	Sign	0.6857 ± 0.0248	0.420	1.633	Table 4
ML-DSA-65	Verify	0.1197 ± 0.0026	0.250	0.479	Table 4

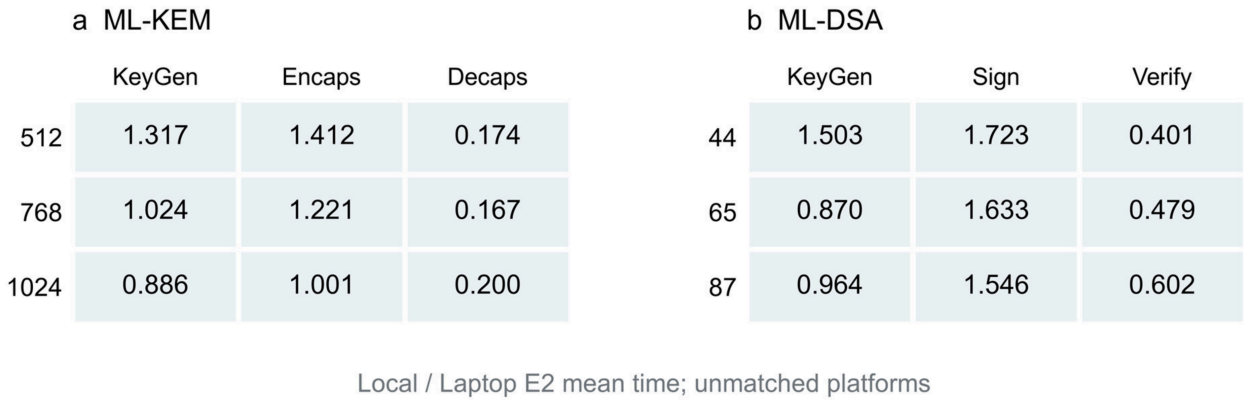


Figure 10. All 18 local-to-Abbasi Laptop E2 mean-time ratios, grouped into (**a**) ML-KEM and (**b**) ML-DSA. Cells retain every algorithm-operation tuple. These are descriptive ratios between unmatched platforms and timing protocols, not causal implementation speedups.

The comparison exposes operation-dependent disagreement rather than a uniform platform factor. Local ML-KEM decapsulation means are much lower than the published Laptop E2 values, whereas several key-generation and signing means are higher. Multiplying every external value by a processor-frequency ratio would not resolve this pattern. Library version, API boundary, compiler, operation semantics, batch design, and system state all remain plausible contributors. The useful outcome is methodological: external studies provide context and a target for replication, but same-host implementation claims must be based on local aligned measurements.

4.8. Security Targets and Non-Equivalent Scales

Table 14 reports classical bit estimates and NIST categories without treating them as one numeric axis. A NIST category is defined by attack-cost comparisons and requirements, not by a promise that every scheme provides exactly the category number transformed into bits. Aigis source-paper estimates are discussed separately because they are not NIST category assignments. See Figure 11.

The figure intentionally uses separate panels. Classical algorithms vulnerable to Shor's algorithm remain unsuitable for long-term post-quantum confidentiality even when their present classical bit estimates look similar to a PQC category label. Future cryptanalysis may change parameter confidence. Algorithm agility and hybrid deployment are therefore risk controls, not admissions that a standardized algorithm is presently broken.

Table 14. Theoretical security targets and interpretation boundaries.

Algorithm/Parameter Set	Family	Reported Target	Underlying Problem	Interpretation
RSA-2048	Classical	112 estimated classical bits	integer factorization	Classical estimate; not post-quantum secure
RSA-3072	Classical	128 estimated classical bits	integer factorization	Classical estimate; not post-quantum secure
ECDH/ECDSA P-256	Classical	128 estimated classical bits	elliptic-curve discrete logarithm	Classical estimate; not post-quantum secure
X25519/Ed25519	Classical	128 estimated classical bits	elliptic-curve discrete logarithm	Classical estimate; not post-quantum secure
ML-KEM-512/ML-DSA-44	Lattice	NIST Category 1	Module-LWE/Module-SIS	NIST category; not a continuous bit-equivalent scale
ML-KEM-768/ML-DSA-65	Lattice	NIST Category 3	Module-LWE/Module-SIS	NIST category; not a continuous bit-equivalent scale
ML-KEM-1024/ML-DSA-87	Lattice	NIST Category 5	Module-LWE/Module-SIS	NIST category; not a continuous bit-equivalent scale
Falcon-512/FN-DSA	Lattice signature	NIST Category 1	NTRU lattice problems	Category claim subject to the FN-DSA standardization status
Classic McEliece-348864	Code-based	NIST Category 1	syndrome decoding	NIST category; not a continuous bit-equivalent scale

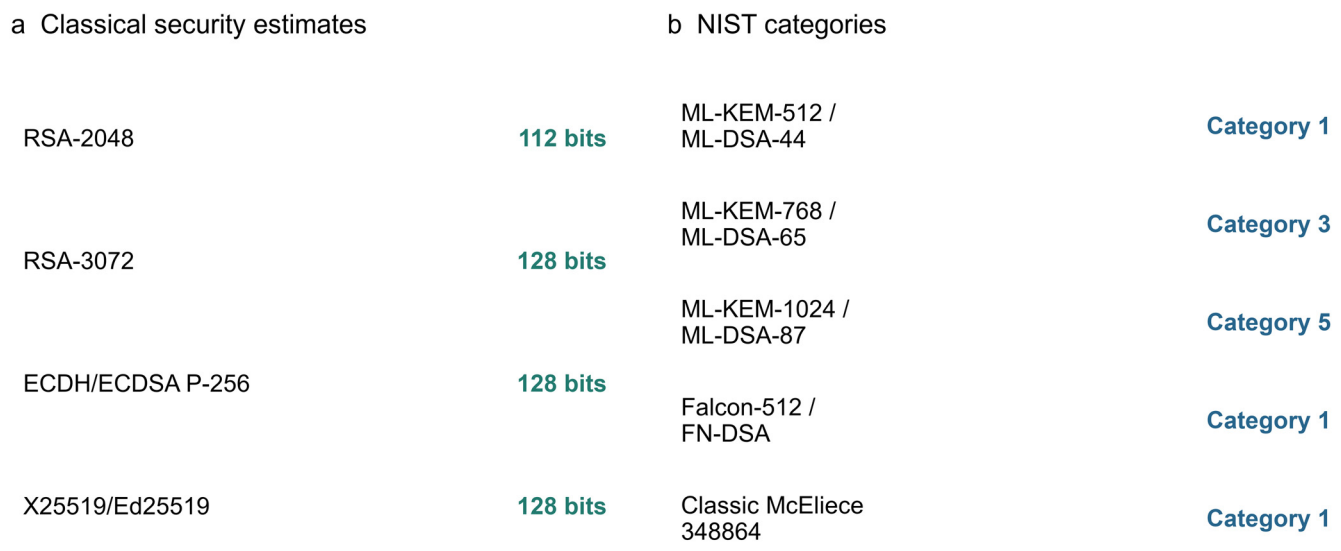


Figure 11. (**a**) Classical security-bit estimates and (**b**) NIST category labels. The categorical listings avoid a common bar-length scale: category numbers are not continuous security-bit equivalents. Values are theoretical context, not measured security.

4.9. Official Standards-Vector Conformance

Table 15 reports 690 implementation-vector results: 345 selected projected cases executed independently through each library adapter. Public keys, secret keys, ciphertexts, shared secrets, and deterministic signatures matched the expected projected bytes; verification returned the expected Boolean result. NIST's ACVP documentation defines the protocol context [33], while Supplementary Table S1 identifies the five liboqs projection files, pinned commit, checkout paths, and projection-file hashes used here. Record-level

projection names, group/case identifiers, expected/actual hashes, and outcomes are included in the supplementary source-data file. The archive now includes the complete projections, selected inputs, expected and actual output bytes, and their hashes. The acquisition script identifies the upstream release, but the original downloaded archive and request/response files were not retained. These records permit byte-level auditing of the recorded outcomes; they are not an independently executed validation run. The result is selected-vector evidence, not certification, exhaustive ACVP coverage, or evidence about side channels and protocol identifiers.

Table 15. Selected FIPS 203/204 ACVP results by operation and implementation.

Operation	Implementation	Cases	Pass	Comparison Contract
kem-keygen	liboqs	75	75	Exact bytes
kem-keygen	pqmagic	75	75	Exact bytes
kem-encaps	liboqs	75	75	Exact bytes
kem-encaps	pqmagic	75	75	Exact bytes
kem-decaps	liboqs	30	30	Exact bytes
kem-decaps	pqmagic	30	30	Exact bytes
sig-keygen	liboqs	75	75	Exact bytes
sig-keygen	pqmagic	75	75	Exact bytes
sig-sign	liboqs	45	45	Exact bytes
sig-sign	pqmagic	45	45	Exact bytes
sig-verify	liboqs	45	45	Expected Boolean result
sig-verify	pqmagic	45	45	Expected Boolean result

4.10. Full-Parameter Bidirectional Interoperability

Across 12 interoperability matrix cells, covering six parameter sets and two cross-library directions per set, the experiment executed 100 deterministically generated cases per cell. For ML-KEM, one library generated the key pair, the other encapsulated to that public key, and the key-generating library decapsulated the ciphertext. A case passed when both sides derived the same shared secret. For ML-DSA, one library generated the key pair and signature and the other verified the signature with the same message and context. All 1200 cases passed within this raw-artifact test scope. The result is not an estimate of field failure probability and does not cover untested protocols, releases, or platforms. Table 16 reports all bidirectional paths, and Figure 12 visualizes the passed/executed counts.

Table 16. Bidirectional cross-implementation positive cases. Arrows indicate the producer-to-consumer operation sequence.

Parameter Set	Cross-Implementation Test Path	Executed Cases	Passed Cases
ML-KEM-512	PQMagic KeyGen → liboqs Encaps → PQMagic Decaps	100	100
ML-KEM-512	liboqs KeyGen → PQMagic Encaps → liboqs Decaps	100	100
ML-KEM-768	PQMagic KeyGen → liboqs Encaps → PQMagic Decaps	100	100
ML-KEM-768	liboqs KeyGen → PQMagic Encaps → liboqs Decaps	100	100
ML-KEM-1024	PQMagic KeyGen → liboqs Encaps → PQMagic Decaps	100	100
ML-KEM-1024	liboqs KeyGen → PQMagic Encaps → liboqs Decaps	100	100
ML-DSA-44	PQMagic Sign → liboqs Verify	100	100
ML-DSA-44	liboqs Sign → PQMagic Verify	100	100
ML-DSA-65	PQMagic Sign → liboqs Verify	100	100
ML-DSA-65	liboqs Sign → PQMagic Verify	100	100

Table 16. *Cont.*

Parameter Set	Cross-Implementation Test Path	Executed Cases	Passed Cases
ML-DSA-87	PQMagic Sign → liboqs Verify	100	100
ML-DSA-87	liboqs Sign → PQMagic Verify	100	100

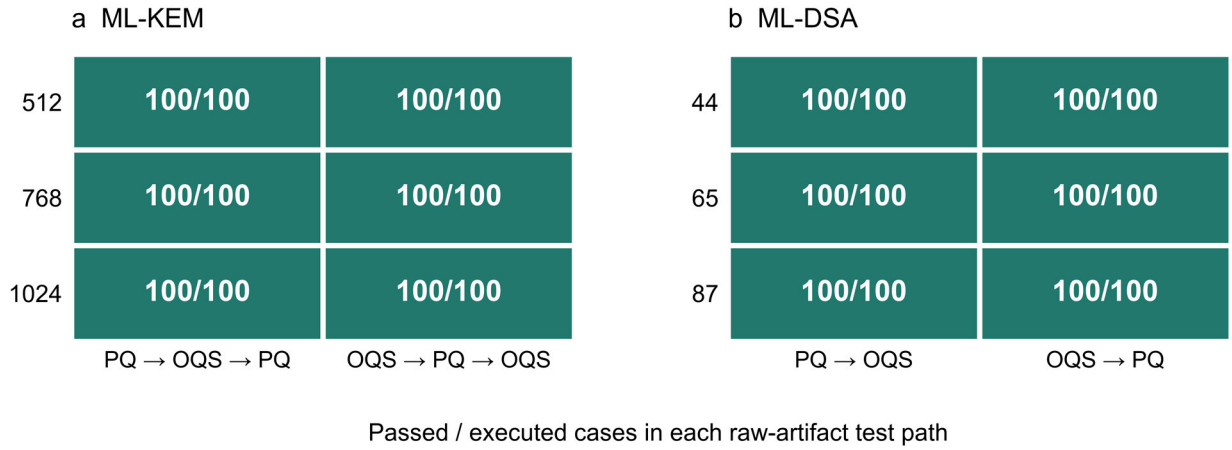


Figure 12. All twelve applicable bidirectional test cells for (**a**) ML-KEM and (**b**) ML-DSA, with passed/executed counts. PQ denotes PQMagic-SHAKE and OQS denotes liboqs. KEM paths list KeyGen → Encaps → Decaps providers; signature paths list signing → verification providers. Each cell contains 100 cases. Structurally inapplicable cross-family cells are omitted, not counted as failures.

4.11. Negative Cases and Compatibility Boundaries

All 90 deterministic one-bit mutations met their specified outcome. The 18 ML-KEM ciphertext cases covered both libraries and the first, middle, and last byte of each parameter set; repeated decapsulation returned a stable implicit-rejection secret distinct from the valid secret. The 72 signature-domain cases used the same three-offset policy for messages, signatures, public keys, and contexts and required verification failure. Thirteen PQMagic-SM3 and Aigis configurations are recorded as 'NOT_APPLICABLE' because no aligned liboqs counterpart exists. This deliberately sparse design establishes only the specified rejection behavior. It is not a malformed-length, truncation, noncanonical-encoding, structural-mutation, or fuzzing campaign. Table 17 separates mutation coverage from not-applicable configuration boundaries, and Figure 13 visualizes these evidence layers.

Table 17. Negative-case coverage and separate-configuration boundaries.

Configuration	Mutation or Boundary	Rows	Result
liboqs	ciphertext	9	9
pqmagic	ciphertext	9	9
liboqs	message	9	9
liboqs	signature	9	9
liboqs	public_key	9	9
liboqs	context	9	9
pqmagic	message	9	9
pqmagic	signature	9	9
pqmagic	public_key	9	9
pqmagic	context	9	9
PQMagic-SM3/Aigis	Compatibility boundary	13	NOT_APPLICABLE

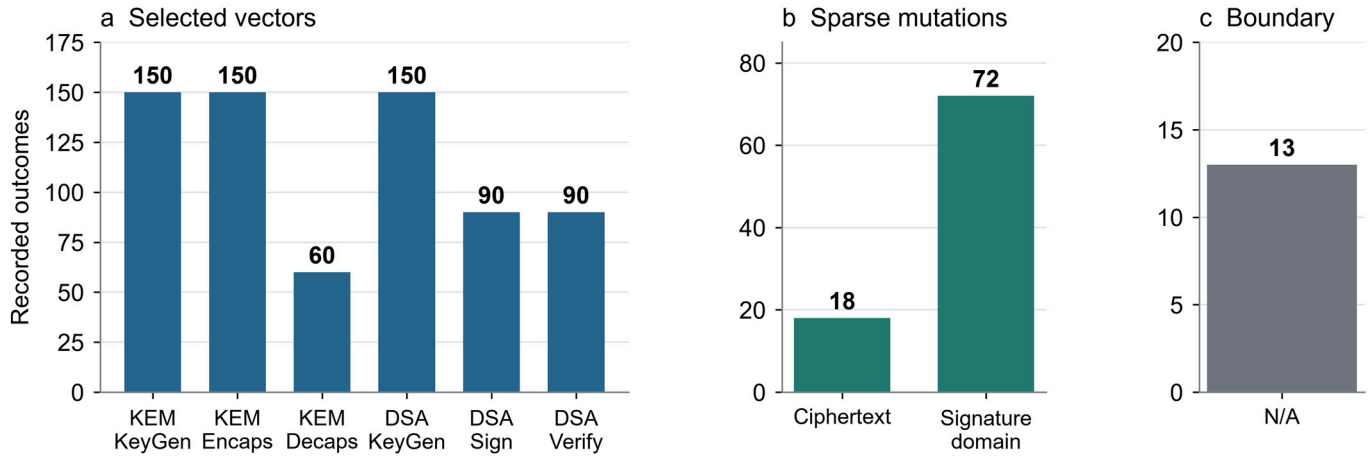


Figure 13. Evidence layers with separate counts: (**a**) 690 selected implementation-vector matches, (**b**) 18 ciphertext and 72 signature-domain mutations, and (**c**) 13 not-applicable configuration boundaries. The mutation set is deliberately sparse, not a malformed-input or fuzzing campaign. Boundary rows are not rejection tests.

4.12. Supporting Same-Commit Cross-Build Replay

All 12 same-commit replay rows passed, covering six parameter sets in both producer–consumer directions. KEM consumers used persisted secret keys and ciphertexts to recover the expected shared secret. Signature consumers verified persisted public keys, messages, contexts, and signatures. In Table 18, Baseline and Current name the archived and clean builds of one pinned PQMagic commit, not different releases. Their binary-hash prefixes distinguish the recorded binaries. This minor regression check shows that the tested build pair consumed the recorded artifacts. It does not test API or encoding evolution, release upgrades, rollback across versions, or arbitrary build options. The supplied supplementary manifests retain complete recorded binary hashes; both recorded DLLs and all 66 referenced replay payloads are now included and hash-matched. The native harness is also archived. These recovered files support audit of the recorded test inputs and outputs; execution in an independent environment has not been verified during this revision. Figure 14 visualizes the 12 passed/executed replay cells.

Table 18. Cross-build artifact replay between archived and clean PQMagic binaries from one pinned source commit. The arrow denotes producer build → consumer build.

Parameter Set	Build Direction	Pass	Producer Hash Prefix	Consumer Hash Prefix
ML-KEM-512	Baseline → Current	1	381842658b42	c99f1eb5ec28
ML-KEM-512	Current → Baseline	1	c99f1eb5ec28	381842658b42
ML-KEM-768	Baseline → Current	1	381842658b42	c99f1eb5ec28
ML-KEM-768	Current → Baseline	1	c99f1eb5ec28	381842658b42
ML-KEM-1024	Baseline → Current	1	381842658b42	c99f1eb5ec28
ML-KEM-1024	Current → Baseline	1	c99f1eb5ec28	381842658b42
ML-DSA-44	Baseline → Current	1	381842658b42	c99f1eb5ec28
ML-DSA-44	Current → Baseline	1	c99f1eb5ec28	381842658b42
ML-DSA-65	Baseline → Current	1	381842658b42	c99f1eb5ec28
ML-DSA-65	Current → Baseline	1	c99f1eb5ec28	381842658b42
ML-DSA-87	Baseline → Current	1	381842658b42	c99f1eb5ec28
ML-DSA-87	Current → Baseline	1	c99f1eb5ec28	381842658b42

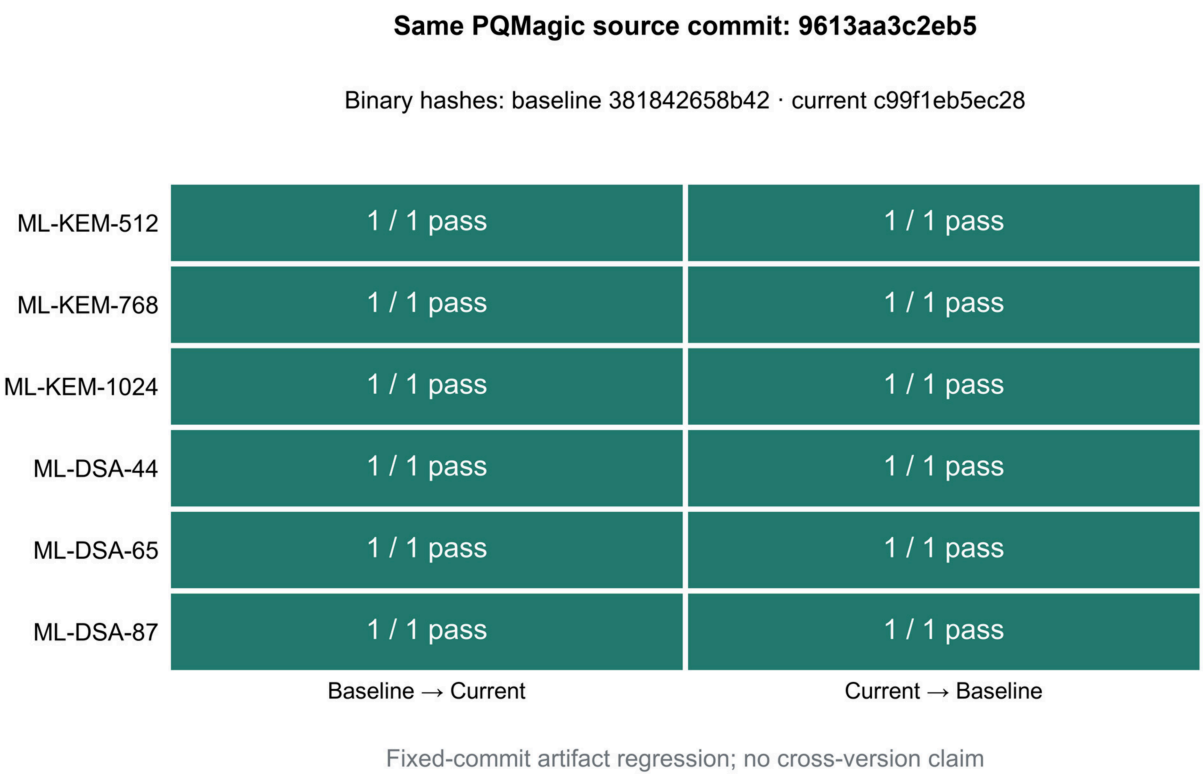


Figure 14. All twelve same-commit PQMagic replay outcomes, grouped by parameter set and build direction. Binary-hash prefixes identify the recorded build pair. Each cell is one passed/executed case; this is a fixed-commit artifact regression check, not cross-version evidence.

5. Deployment Considerations: A Taxonomy and Checklist

5.1. Five Categories of Considerations

Table 19 organizes computation, time, storage, network, and deployment considerations into a taxonomy and checklist. It is not a deployment decision method. No weights, aggregate score, universal acceptance thresholds, or validation against real deployments are provided. The categories identify which evidence is available and which additional measurements a deployment review requires. Application-specific operation frequencies, capacity limits, protocol behavior, and update requirements must be supplied separately. Compatibility checks inform that review but are not sufficient deployment eligibility criteria.

Computation and time are related but not identical. Computation describes dominant arithmetic and acceleration opportunities; time is the observed outcome under a platform and workload. An NTT-oriented implementation may benefit from SIMD, but a measured signing path may still be dominated by sampling or hashing. Storage includes long-lived keys, temporary workspaces, and cached peer material. This study measures API artifact sizes, not native peak workspace, so memory-capacity decisions require a separate process-level or allocator-level experiment.

Network cost begins with cryptographic artifacts but ends at protocol traces. Public keys, ciphertexts, and signatures are necessary inputs to a payload model. Certificates add identifiers, extensions, issuer signatures, chain elements, and encoding. Protocols add framing, negotiation, retransmission, and fragmentation. Consequently, Equation (13) is useful for ordering rough capacity but not for claiming a measured handshake size.

Deployment includes standardization, implementation provenance, build reproducibility, algorithm identifiers, key metadata, rollback, and regression testing. PQMagic's value is not reducible to speed. It offers domestic maintenance, SM3 compatibility, and

implementation diversity. liboqs offers broad international algorithm coverage and integration experience. A governed dual-track design may use standardized algorithms as an interoperability baseline and retain a domestic implementation or Aigis path for controlled environments.

Table 19. Taxonomy of deployment considerations and questions requiring additional evidence.

Dimension	Primary Evidence	Checklist Question	Evidence Boundary/Further Work
Computation	Measured operation time and theoretical arithmetic	Can the workload meet latency/throughput targets?	Specify operation mix; no aggregate decision score is validated
Storage	Measured key and output sizes	Can endpoints and key stores hold the material?	API sizes only; measure native workspace and storage overhead separately
Time	Mean, P95, CI, and cross-session sensitivity	Is the timing baseline repeatable enough for regression use?	Snapshot only; replicate sessions before setting regression thresholds
Network	Public key plus ciphertext/signature bytes	Will handshakes, certificates, or updates fragment?	Payload estimates only; measure protocol traces and fragmentation
Deployment	Standards status, backend, implementation source, correctness gates	What additional evidence is needed for integration and release migration?	Require protocol and cross-version tests; same-commit replay is insufficient

Compatibility evidence is a necessary input to deployment review, not an automatic eligibility gate or performance-ranking rule. Reviewers of a candidate should document its standard profile, identifiers, selected-vector outcomes, tested producer–consumer directions, and persisted-artifact requirements. Passing these checks does not establish protocol-level replaceability or safe release migration. The same-commit replay result cannot satisfy a cross-version requirement. PQMagic-SM3 and Aigis remain separate configurations requiring their own identifiers, ecosystem support, and integration tests.

5.2. Illustrative Workload Estimates

Let f_o be the expected number of calls to operation o in a workload interval and let t_o be its measured mean. A first-order compute-time model is

$$T^{est}_{workload} = \sum_{o \in \mathcal{O}} f_o \, t_o. \tag{14}$$

This derived estimate illustrates one checklist item rather than combining the five categories into a decision score. It assumes that operation means transfer to the workload, ignores contention, and excludes protocol and queueing delay. The single-session means do not validate that transfer. A corresponding artifact estimate is

$$B^{est}_{workload} = f_{pk} S_{pk} + f_{ct} S_{ct} + f_{sig} S_{sig} + B_{protocol}. \tag{15}$$

The term $B_{protocol}$ must be measured or modeled separately. These equations show why an aggregate three-operation sum can be misleading. If a server generates one key per day and decapsulates one million ciphertexts, decapsulation dominates. If a code-signing authority signs infrequently and millions of devices verify, verification and signature transport dominate.

5.3. Agility and Future Regression Checks

Cryptographic agility is the ability to inventory, replace, test, and retire cryptographic mechanisms without uncontrolled disruption. NIST guidance treats this as a lifecycle practice involving policies, dependencies, interoperability, testing, and transition [34]. The benchmark contributes a measurement baseline to that lifecycle. It does not measure governance effectiveness.

For each deployed tuple $(library, version, algorithm, parameter, backend, operation)$, an organization can store a baseline mean, P95, correctness vector, and artifact size. An illustrative, unvalidated regression indicator compares a new mean $\overline{x}_{new}$ with a baseline $\overline{x}_{base}$ using

$$\Delta = \frac{\overline{x}_{new} - \overline{x}_{base}}{\overline{x}_{base}}. \tag{16}$$

No threshold for Δ is calibrated or validated here. A deployment-specific threshold would require independent session replication, a stated tolerance, and a plan for investigating changes. A threshold exceedance alone would not prove a defective library. Key formats, negotiation behavior, and rollback across releases require separate integration and cross-version tests. The same-commit replay check does not validate those lifecycle requirements.

5.4. Scenario Viewpoints

Table 20 illustrates questions for deployment review rather than validated choices. Standardized ML-KEM and ML-DSA provide a starting point for interoperability testing, but standard status alone does not demonstrate protocol integration. PQMagic, Aigis, and SM3 configurations add implementation or policy considerations outside the aligned cross-library claim. Large artifacts motivate provisioning and transport measurements, while hybrid designs require explicit composition, negotiation, identifier, and downgrade checks. None of these viewpoints has been validated against a real deployment in this study.

Table 20. Illustrative deployment viewpoints and limitations.

Scenario	Illustrative Starting Point	Why	Condition or Caveat
Interoperable Internet-facing key establishment	ML-KEM-768 in a standards-oriented stack	Standard status and balanced size/performance	Protocol and certificate behavior still require real integration tests
Controlled domestic infrastructure	PQMagic ML-KEM/ML-DSA or Aigis with optional SM3	Domestic maintenance path and SM3 compatibility	Enable SM3 selectively because overhead is operation dependent
High-volume signing service	Benchmark ML-DSA implementation and parameter set against actual sign/verify mix	PQMagic-SHAKE signing was faster locally	Key protection and side-channel controls dominate production acceptance
Firmware verification	ML-DSA or compact-signature alternative	Verification and artifact size matter more than key generation	Long-lived verifier updates must support algorithm replacement
Constrained links	Prefer moderate artifacts; avoid unmodeled certificate expansion	Fragmentation can dominate primitive latency	Use measured protocol traces before deployment
Algorithm-diversity requirement	Hybrid or dual-track implementation	Reduces dependence on one mathematical family or supplier	Composition, negotiation, downgrade resistance, and lifecycle management must be specified

The checklist combines measured artifacts, theoretical security context, and software-deployment considerations without aggregating them into a score. It is illustrative guidance, not a certification, validated decision rule, or product recommendation.

6. Discussion

6.1. Primary Contribution and Supporting Observations

The central contribution is the auditable link between selected-vector conformance, bidirectional raw-artifact interoperability, and deterministic negative behavior for the pinned ML-KEM/ML-DSA implementations. Supporting timing and backend observations remain separate from that compatibility evidence. The recorded ML-KEM ratios were not uniformly favorable to PQMagic-SHAKE, and ML-DSA signing differences remain session-bounded. SM3 mean-time ratios varied by operation and parameter set. These supporting results identify candidates for retesting; they neither establish stable rankings nor extend conformance claims to Aigis or SM3 configurations.

The evidence taxonomy also changes the interpretation of familiar metrics. A library-reported public-key length is Measured at the API boundary. Throughput is derived from a timing mean. The PQC certificate value is estimated. $O(n\log n)$ is theoretical. Mixing them without labels can create false precision. With labels, each quantity can still be useful while retaining its inference limit.

6.2. Single-Session Interpretation and Temporal Sensitivity

The primary timing data are a descriptive 50-observation snapshot, not a multi-session estimate of performance. Comparison with the earlier 30-observation run reveals temporal sensitivity across 78 matched algorithm-operation-configuration rows. The median ratio of the 50-observation mean to the 30-observation mean was 1.160. The maximum was 3.052 for PQMagic-SHAKE Aigis-Sig-1 signing, whereas PQMagic-SHAKE ML-DSA-87 signing had a ratio of 0.529. Thus, the observed change was not a common multiplier across all rows. These 78 rows are different configurations and operations, not 78 independent sessions. The runs were not pooled because system state and execution order were not controlled as between-session factors.

Equation (11) is retained as a compact, consistently computed summary of the mean because each recorded operation used 50 adaptive-batch observations and the existing workbooks report the corresponding standard deviation. It is not used for hypothesis testing, equivalence, causal implementation effects, or cross-session inference. Skewness and serial correlation can invalidate its nominal coverage, so the interval must be read alongside the retained median, minimum, maximum, and P95. The available two-run sensitivity comparison cannot determine whether temporal drift is systematic by algorithm, operation, or backend. We therefore make no claim of stable superiority or equivalence and do not estimate between-session variance from within-session batches. Resampling batches from one run would not replace independent session replication. A follow-up should randomize or interleave execution order across independent days, log power and thermal conditions, and replicate each configuration before applying a session-aware bootstrap or hierarchical model. The supplementary v13 workbooks expose the descriptive summaries used here; individual raw v13 batches are unavailable. A separately labeled v14 workbook contains 4800 raw observations from a later session and is not substituted for the main results.

6.3. Security and Implementation Risk

No benchmark can establish cryptographic security. Standard algorithms rely on public analysis, parameter selection, and reductions; implementations add memory safety,

side-channel, randomness, fault, and update risks. Aigis source results are important evidence [27], but source-paper dimensions matching an API do not guarantee that every later implementation path is covered by the same proof. Conversely, lack of NIST standard status does not imply an algorithm is broken; it changes the governance and interoperability burden.

Future cryptanalysis may alter confidence in a lattice, code, or hash construction. Hybrid use combines independent secrets or signatures to reduce dependence on one family, but composition must be specified. For KEMs, a combiner such as $K = KDF(K_1 \parallel K_2 \parallel context)$ is preferable to an informal concatenation with undefined context. Negotiation must resist downgrade, and keys must record which mechanisms created them. The system must also define when one component can be removed. This lifecycle requirement is why agility is a deployment property, not an algorithm property.

6.4. Implementation and Supply-Chain Diversity

The local data do not support a universal performance claim for either library. They support narrower observations: PQMagic exposed working SHAKE and SM3 configurations, the tested Aigis modes passed the stated functional gates, and several ML-DSA operations recorded lower means in the aligned session. The broader value of an additional implementation path is diversity of maintenance, backend support, and supply chain. Those benefits are governance considerations rather than measured cryptographic properties.

Supply-chain diversity has a cost. Two libraries require two sets of test vectors, build pipelines, vulnerability monitoring, API adapters, and rollback procedures. Algorithm diversity adds identifiers and policy. These costs are justified only when the organization can maintain them. A dual-track design without continuous regression and inventory can increase rather than reduce risk.

6.5. Compatibility Evidence Beyond Performance Benchmarking

The primary evidence answers three bounded questions. Selected vector matching tests expected standard outputs, bidirectional producer–consumer cases test the raw-artifact interface, and deterministic mutations test specified negative behavior. Each has an explicit comparison contract and can fail independently. The additional replay check tests recorded persisted artifacts across a fixed-commit build pair and is a minor regression result. These executable checks must be distinguished from Section 5, whose deployment taxonomy remains an organizational checklist without validated scoring, thresholds, or real-deployment evaluation.

The results also delimit what has not been shown. Cross-library success does not prove constant-time execution, cryptographic security, secure randomness, key erasure, or protocol negotiation. Same-commit cross-build success establishes neither compatibility between releases nor safe upgrade or rollback procedures. The all-pass records provide evidence for the specified cases only; they do not establish universal compatibility or deployment readiness.

7. Threats to Validity

Internal validity. Windows scheduling, background processes, dynamic processor frequency, thermal state, cache state, and allocator behavior can affect wall-clock timing. Adaptive batching and 50 repeated observations reduce timer granularity but do not remove autocorrelation or temporal drift. The cross-session comparison confirms that a narrow within-session interval does not imply a stable cross-day mean. The benchmark did not pin cores, disable turbo behavior, or operate in a laboratory power state.

Construct validity. The measured boundary is a native cryptographic call orchestrated from Python. Adaptive batches reduce per-call orchestration cost, but the result is still not a pure instruction-count measurement. Throughput is derived from serial means and is not concurrent service capacity. API artifact sizes omit protocol framing. Native peak memory, CPU utilization, and hardware energy were not measured. Python allocation tracing was intentionally excluded because it would not capture comparable native OpenSSL, liboqs, and PQMagic allocations.

External validity. The evidence comes from one Windows x86_64 host and one build toolchain. It cannot be extrapolated directly to ARM, RISC-V, domestic processors, servers with different vector units, or embedded devices. Abbasi et al. provide external context [7], and the comparison demonstrates rather than removes platform dependence. Future work should repeat the same correctness gates, algorithms, timing boundary, and evidence labels across aligned x86_64, ARM, RISC-V, and domestic processor platforms.

Conclusion validity. The normal-approximation interval may be inaccurate for skewed or correlated timing data. The workbook retains median and P95, but 50 within-session observations do not support detailed tail modeling or between-session inference. All performance comparisons are descriptive, without family-wide significance, equivalence, or stable-ranking claims. The earlier and current runs are not pooled, and their matched rows are not treated as independent session replicates. Future variance estimation requires replicated sessions and a session-aware analysis; bootstrapping individual batches alone would not address temporal drift.

Security validity. Functional gates do not establish IND-CCA or SUF-CMA security, validate randomness, or detect side channels. Security claims are attributed to standards and primary publications [5,6,21,24,27]. The experiment does not certify PQMagic, liboqs, Aigis, or any deployment. It measures stated software behavior under a controlled boundary.

Artifact-estimation validity. PQC certificate rows are illustrative additive estimates, not generated X.509 certificates. The common 300-byte allowance was not calibrated against a certificate corpus and may not represent a particular profile, chain, extension set, or hybrid certificate. Changing the allowance shifts each PQC estimate one-for-one and leaves only their ordering under a common allowance unchanged. Real profile generation, chain construction, and protocol capture are required before absolute certificate or handshake conclusions are made.

These limitations define a focused follow-up. The primary conformance and interoperability tests should be reproduced with aligned interfaces on additional architectures and operating systems. Independent multi-day and multi-compiler sessions are needed to assess performance reproducibility. True cross-version replay and protocol integration must separately test release evolution and encoded deployment artifacts. Concurrency, native process memory, hardware energy, and real-deployment validation of the checklist remain unmeasured rather than implied extensions of the present results.

Conformance selection. The ACVP set is version-pinned and covers the selected pure external ML-DSA interfaces and FIPS 203 operations; it does not exhaust every future vector revision, prehash interface, or external-μ interface. Exact matches are reported at the implementation-vector level, not as a certification claim.

Interoperability and replay scope. The deterministic 100-case cells were executed on one Windows x86_64 host. The two PQMagic replay binaries share a source commit, so their 12 passing cases check persisted-artifact consumption across that build pair only. They do not test semantic evolution, serialization changes, or migration across releases. Real key stores, HSMs, TLS stacks, X.509 encodings, network loss, and policy engines remain outside the experiment. The supplementary ZIP includes replay records, both recorded

DLLs, the native harness, and all referenced binary payloads. Their integrity has been checked, but no independent replay or source rebuild was performed during this revision.

8. Conclusions

This study presents an auditable conformance and raw-artifact interoperability workflow for pinned liboqs and PQMagic-SHAKE implementations of ML-KEM and ML-DSA. The selected ACVP projections produced 690 expected implementation-vector outcomes. All 1200 bidirectional positive cases and 90 deliberately sparse deterministic one-bit mutations met their specified outcomes across the tested parameter sets and interfaces. The negative cases do not constitute malformed-input testing or fuzzing. A minor supporting check passed 12 persisted-artifact replay cases between builds of one PQMagic commit, without establishing cross-version compatibility. Performance, Aigis, and SHAKE/SM3 results remain supporting single-session observations. The five deployment categories form a taxonomy and checklist, not a validated decision framework.

The practical implication is that successful self-tests or favorable timings cannot substitute for explicit producer–consumer testing. The recorded evidence applies only to the selected projections, pinned commits, Windows host, and raw interfaces. Timing and artifact-size observations may inform subsequent deployment review but do not establish stable rankings, protocol compatibility, or deployment eligibility. Priorities for future work are independent reproduction, replicated session-aware benchmarking, true cross-version replay, and X.509/TLS/PKI integration. Hardware-backed storage, side-channel evaluation, and deployment-specific validation remain separate requirements.

Supplementary Materials: The following supporting information can be downloaded at https://www.mdpi.com/article/10.3390/computers15100642/s1: Table S1, ACVP projection provenance, selection rules, paths, versions, and SHA-256 hashes (reproducibility/ACVP_PROVENANCE.csv and .md); Data S1, publication workbooks and positive, negative, ACVP, and replay case records; Data S2, referenced binary artifacts and manifests; Code S1, the native C harness, adapters, preparation, verification, and figure-regeneration scripts; File S1, build-configuration evidence, binary hashes, inventories, and the material-availability statement. The archive is Supplementary_Materials_R7.zip (SHA-256: b498fe9827e77b145e174bf7d375926e76a499c11b17c6775fd6051e9bd2b26f).

Author Contributions: Conceptualization, S.X.; methodology, S.X. and X.L.; software, X.L.; validation, X.L., H.W. and H.Z.; formal analysis, H.W. and H.Z.; investigation, X.L.; resources, S.X. and H.Z.; data curation, X.L.; writing—original draft preparation, X.L.; writing—review and editing, S.X., H.W. and H.Z.; visualization, X.L.; supervision, S.X.; project administration, S.X.; funding acquisition, S.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Yunnan Provincial Science and Technology Plan Project (Major Science and Technology Special Program), grant number 202502AD080015; and the Fundamental Research Funds for the Central Universities, grant number 20230070Z0114.

Data Availability Statement: The evidence is provided in Supplementary_Materials_R7.zip (SHA-256: b498fe9827e77b145e174bf7d375926e76a499c11b17c6775fd6051e9bd2b26f). It contains unchanged publication workbooks and case records, five hash-verified ACVP projections, current and baseline binaries, the native harness and source code, build records including all six Ninja rule files and the recovered static liboqs archive, all 10,146 referenced test payloads, and source data and code for fourteen figures. The read-only verification script checks file hashes and payload lengths without running native code. The v13 and earlier-run individual timing batches and original upstream ACVP requests/responses and downloaded archive were not retained. Full source and toolchain installations remain external rebuild prerequisites; see MISSING_MATERIALS.md. The separately labelled v14 raw session does not replace v13. No independent cryptographic replay, public repository DOI or accession is asserted.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
2. Grover, L.K. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*; ACM: New York, NY, USA, 1996; pp. 212–219. [CrossRef]
3. Chen, L.; Jordan, S.; Liu, Y.-K.; Moody, D.; Peralta, R.; Perlner, R.; Smith-Tone, D. *NISTIR 8105: Report on Post-Quantum Cryptography*; NIST: Gaithersburg, MD, USA, 2016. [CrossRef]
4. Mosca, M. Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Secur. Priv.* **2018**, *16*, 38–41. [CrossRef]
5. National Institute of Standards and Technology. *FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard*; NIST: Gaithersburg, MD, USA, 2024. [CrossRef]
6. National Institute of Standards and Technology. *FIPS 204: Module-Lattice-Based Digital Signature Standard*; NIST: Gaithersburg, MD, USA, 2024. [CrossRef]
7. Abbasi, M.; Cardoso, F.; Vaz, P.; Silva, J.; Martins, P. A practical performance benchmark of post-quantum cryptography across heterogeneous computing environments. *Cryptography* **2025**, *9*, 32. [CrossRef]
8. Souvatzidaki, K.; Limniotis, K. Post-Quantum Key Exchange in TLS 1.3: Further Analysis on Performance of New Cryptographic Standards. *Cryptography* **2025**, *9*, 73. [CrossRef]
9. Raavi, F.; Khan, F.; Wuthier, F.; Chandramouli, P.; Balytskyi, Y.; Chang, S.-Y. Security and Performance Analyses of Post-Quantum Digital Signature Algorithms and Their TLS and PKI Integrations. *Cryptography* **2025**, *9*, 38. [CrossRef]
10. Liu, H.; Chen, L.; Lu, X.; Wang, H.; Bai, L.; Wang, M.; Ren, P. A visual-textual mutual guidance fusion network for remote sensing visual question answering. *Pattern Recognit.* **2026**, *176*, 113258. [CrossRef]
11. Li, X.; Sun, W.; Ji, Y.; Huang, W. A Plot-to-Track Association Framework Based on Graph Representation Learning for Compact HFSWR. *IEEE Trans. Aerosp. Electron. Syst.* **2026**, *62*, 12742–12760. [CrossRef]
12. Dam, D.-T.; Tran, T.-H.; Hoang, V.-P.; Pham, C.-K.; Hoang, T.-T. A Survey of Post-Quantum Cryptography: Start of a New Race. *Cryptography* **2023**, *7*, 40. [CrossRef]
13. Fitzgibbon, G.; Ottaviani, C. Constrained Device Performance Benchmarking with the Implementation of Post-Quantum Cryptography. *Cryptography* **2024**, *8*, 21. [CrossRef]
14. Barker, W.; Polk, W.; Souppaya, M. *Getting Ready for Post-Quantum Cryptography: Exploring Challenges Associated with Adopting and Using Post-Quantum Cryptographic Algorithms*; NIST Cybersecurity White Paper 15; NIST: Gaithersburg, MD, USA, 2021. [CrossRef]
15. Ahmed, N.; Zhang, L.; Gangopadhyay, A. A Survey of Post-Quantum Cryptography Support in Cryptographic Libraries. In *Proceedings of the 2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*; IEEE: New York, NY, USA, 2025; pp. 906–917. [CrossRef]
16. Paquin, C.; Stebila, D.; Tamvada, G. Benchmarking Post-quantum Cryptography in TLS. In *Post-Quantum Cryptography*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2020; pp. 72–91. [CrossRef]
17. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of STOC 2005*; ACM: New York, NY, USA, 2005; pp. 84–93. [CrossRef]
18. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*; Springer: Berlin, Germany, 2010; pp. 1–23. [CrossRef]
19. Albrecht, M.R.; Player, R.; Scott, S. On the concrete hardness of learning with errors. *J. Math. Cryptol.* **2015**, *9*, 169–203. [CrossRef]
20. Fujisaki, E.; Okamoto, T. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO 1999*; Springer: Berlin, Germany, 1999; pp. 537–554. [CrossRef] [PubMed]
21. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehle, D. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy*; IEEE: Piscataway, NJ, USA, 2018; pp. 353–367. [CrossRef]
22. Hofheinz, D.; Hovelmanns, K.; Kiltz, E. A modular analysis of the Fujisaki–Okamoto transformation. In *TCC 2017*; Springer: Cham, Switzerland, 2017; pp. 341–371. [CrossRef]
23. Jiang, H.; Zhang, Z.; Chen, L.; Wang, H.; Ma, Z. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In *CRYPTO 2018*; Springer: Cham, Switzerland, 2018; pp. 96–125. [CrossRef]
24. Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 238–268. [CrossRef]
25. National Institute of Standards and Technology. *FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*; NIST: Gaithersburg, MD, USA, 2015. [CrossRef]
26. *GB/T 32905-2016*; Information Security Technology–SM3 Cryptographic Hash Algorithm. Standards Press of China: Beijing, China, 2016.

27. Zhang, J.; Yu, Y.; Fan, S.; Zhang, Z.; Yang, K. Tweaking the asymmetry of asymmetric-key cryptography on lattices: KEMs and signatures of smaller sizes. In *Public-Key Cryptography–PKC 2020*; Springer: Cham, Switzerland, 2020; pp. 37–65. [CrossRef]
28. Chen, M.-S.; Chou, T. Classic McEliece on the ARM Cortex-M4. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2021**, *2021*, 125–148. [CrossRef]
29. National Institute of Standards and Technology. *FIPS 205: Stateless Hash-Based Digital Signature Standard*; NIST: Gaithersburg, MD, USA, 2024. [CrossRef]
30. Open Quantum Safe. liboqs: C Library for Quantum-Resistant Cryptographic Algorithms. Available online: https://github.com/open-quantum-safe/liboqs (accessed on 16 July 2026).
31. PQCrypto. PQMagic: Post-Quantum Cryptographic Algorithm Library. Available online: https://gitee.com/pqcrypto/pqmagic (accessed on 16 July 2026).
32. Stebila, D.; Mosca, M. Post-quantum key exchange for the Internet and the Open Quantum Safe project. In *Selected Areas in Cryptography–SAC 2016*; Springer: Cham, Switzerland, 2017; pp. 14–37. [CrossRef]
33. National Institute of Standards and Technology. Automated Cryptographic Validation Protocol Documentation. Available online: https://pages.nist.gov/ACVP/ (accessed on 9 September 2026).
34. Barker, E.; Chen, L.; Cooper, D.; Moody, D.; Regenscheid, A.; Souppaya, M.; Newhouse, W.; Housley, R.; Turner, S.; Barker, W.C.; et al. *Considerations for Achieving Crypto Agility: Strategies and Practices*; NIST: Gaithersburg, MD, USA, 2025. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.