Binding Post-Quantum Cryptography to Copyleft: The Quantum General Public License v3.0 and an Empirical Evaluation of Mandatory Quantum-Safe Distribution Integrity

Šarūnas Grigaliūnas, *Member, IEEE*

***Abstract*—The post-quantum transition is usually framed as a *confidentiality* problem—"harvest now, decrypt later." Its *integrity* dual is less discussed but structurally identical: a classical asymmetric signature attached to a released dataset today is forgeable by a future cryptographically-relevant quantum computer, after which an adversary can substitute the artifact and any downstream verifier will accept the substitution. For open science, whose archival horizons routinely exceed the security horizon of the primitives that protect distributions at release, this is a live threat. We introduce the Quantum General Public License (QGPL) v3.0—to our knowledge the first copyleft license to make post-quantum (PQ) distribution integrity a *license condition* rather than a recommendation—and we evaluate it empirically. The construction binds NIST-standardized PQ signatures (FIPS 204 ML-DSA, FIPS 205 SLH-DSA, and the stateful schemes of NIST SP 800-208) into the viral grant of a copyleft license, so that a single upstream issuer's choice of license propagates a PQ integrity floor through the entire downstream reuse graph. We specify a companion Compliance Toolkit: a JCS-canonicalized JSON manifest, a two-tier offline-root key hierarchy that needs no certificate authority, and a strictly *additive* composite encoding for hybrid PQ+classical signatures. We then build a spec-faithful reference using real FIPS 204/205 primitives and run three evaluation scenarios on commodity hardware: (A) per-release cryptographic cost across the conformant signature surface; (B) metadata overhead at release scale; and (C) verification, conformance negative-case detection, and long-horizon archival recovery. ML-DSA-65 signs a release in 0.72 ms and verifies in 0.37 ms; a hybrid adds only 64 bytes and 0.2 ms; mandatory metadata falls below 1% of payload for artifacts $\geq$ 11 KiB and amortizes to 148 bytes per artifact in a thousand-file aggregate manifest; all six conformance negative cases are detected, and a hybrid release whose classical half is later broken remains recoverable through a PQ-only verification mode. We discuss the OSI/FSF approval tradeoff, the enforceability of conditions over factual data, and the hardware-attestation gap, which we mitigate through structured operational disclosure rather than cryptographic proof.**

***Index Terms*—Post-quantum cryptography, copyleft, software supply chain, distribution integrity, digital signatures, ML-DSA, SLH-DSA, hybrid signatures, research data, FAIR, crypto-agility.**

I. Introduction

THE "harvest now, decrypt later" (HNDL) framing for the post-quantum transition has focused public discussion on **confidentiality**: encrypted traffic captured today can be decrypted by a future cryptographically-relevant quantum computer (CRQC) [1–4], so transport channels need PQ protection now, before the threat materializes. The HNDL model is by now well characterized as a time-dependent risk whose exposure window is a function of data longevity and migration readiness [5–7], and enterprise migration studies place realistic transition timelines well beyond a decade for large organizations [8, 9].

The parallel **integrity** problem is less discussed but has the same structure. A classical asymmetric signature attached to a distributed dataset today—a Sigstore signature with Ed25519, or a GPG signature over a software release—is forgeable by a future CRQC. Any party who can forge the signature can substitute the dataset, and downstream verifiers who later check the signature against the published key will accept the substitution. Scientific reproducibility, audit trails, supply-chain provenance, and the integrity of the regulatory record become as exposed as encrypted traffic, with a longer time horizon: archived datasets are routinely re-examined decades after release [10–12]. Because Grover's algorithm gives only a quadratic speedup against preimage search [13], the *hash* that binds an artifact to its signature remains adequate at 256 bits; it is the *asymmetric* signature that fails, and exactly that primitive is what every mainstream distribution-integrity system signs with today.

Existing distribution-integrity systems—Sigstore [14], in-toto [15], SLSA [16], the Software Bill of Materials (SBOM) ecosystem [17, 18], and the release-signing conventions of package managers—are uniformly **classical** in their signing primitives. Existing copyleft and open-data

Š. Grigaliūnas is with the Department of Computer Science, Kaunas University of Technology, Kaunas, Lithuania, and with The Open Reality Initiative (e-mail: sarunas.grigaliunas@ktu.lt; ORCID 0000-0001-9268-9244).

Reference implementation, conformance fixture, experiment harness, and the source of this paper are available at https://github.com/Grigaliunas/qgpl under QGPL v3.0.

Manuscript prepared June 13, 2026.

licenses—GPL [19], AGPL, the Creative Commons family [20], the Open Database License [21]—are uniformly **silent** on cryptographic distribution. There is no license condition today that requires a redistributor to ship a PQ-signed artifact, none that propagates such a requirement virally through derivative works, and none that ties license grant to verifiable integrity at all. Without such a binding, voluntary PQ adoption in open science depends on social pressure and the availability of toolchains—both currently weak.

We propose that the *license* is the right binding point. A copyleft grant already propagates conditions through reuse chains: a derivative of a GPL work must also be GPL, transitively, by virtue of the original grant's terms; the viral mechanism is precisely what makes license incompatibilities propagate through dependency graphs in practice [22, 23]. The same mechanism can carry a PQ integrity requirement. If every redistributor of a QGPL-licensed Work must ship Quantum-Safe Distribution Metadata with each distribution, and if downstream modifications must also be released under QGPL, then a single upstream issuer's choice of license propagates a PQ integrity floor through the entire downstream graph. No certificate authority, no central registry, no opt-in toolchain: the license itself carries the requirement, and the legal grant terminates on non-compliance.

A license condition is, however, only as credible as the cost of complying with it. A position paper can assert that mandatory PQ metadata is "cheap"; an engineering paper must show it. We therefore pair the construction with a spec-faithful reference implementation built on real FIPS 204/205 primitives and measure what the construction actually costs.

Contributions.

1) **A legal-technical construction** (§IV) that places mandatory PQ distribution integrity inside a copyleft license's conditions, using the viral clause to propagate the requirement through reuse chains.
2) **A no-infrastructure trust model** (§V)—a two-tier offline-root key hierarchy published in a static, PQ-signed `keys.json`—that delivers rotation and revocation without a certificate authority, a transparency log, or any centralized identity provider.
3) **A strictly additive composite encoding** for hybrid PQ+classical signatures (§V) that forbids threshold semantics, so that adding a classical signature alongside a PQ one only *strengthens* the integrity claim and never substitutes for it.
4) **An empirical evaluation** (§VIII) of a spec-faithful reference across three scenarios—per-release cost, metadata overhead at scale, and verification / conformance / archival recovery—with all numbers produced by really signing and verifying, plus a conformance fixture (§X) that makes the construction implementable and reviewable today.

We discuss tradeoffs honestly in §XIII: the construction may face OSI/FSF approval friction as a field-of-use restriction; copyleft conditions over uncopyrightable factual data have legal limits; and the toolkit cannot cryptographically attest that an offline root key was actually generated offline—a gap we mitigate through structured disclosure, not proof.

A. Why open science is the load-bearing case

The construction is general, but it is sharpest where the gap between archival and cryptographic horizons is widest, and that is open scientific data. Three properties coincide there. First, *longevity*: a reference genome, an astronomical catalog, or a particle-physics dataset is cited and re-analyzed for decades, so a signature that becomes forgeable in 2035 retroactively unprotects data that the community will still be using in 2055. Second, *reuse and derivation*: open datasets are forked, subsetted, merged, and re-released, which is exactly the reuse graph a copyleft condition is built to traverse—and exactly where a voluntary, per-producer convention leaks, because any one redistributor who omits the metadata breaks the chain. Third, *decentralization without a trusted intermediary*: science has no universal CA and no single transparency log it agrees to trust across institutional and national boundaries, so a verification model that needs neither is the only one that travels. A redistributor who strips integrity metadata from a QGPL dataset is in the same posture as one who strips the license—they have lost the grant. The same three properties make classical alternatives a poor fit: Sigstore's OIDC identities and transparency log presuppose live infrastructure that an archival verifier in 2055 cannot assume, and a discretionary signing convention has no mechanism to propagate through the reuse graph at all. The remainder of the paper develops the construction (§IV–VI), then measures it (§VII–VIII) to show the integrity layer it adds is, in every dimension a publisher or recipient pays for, cheap.

II. Background

A. NIST post-quantum standardization

Between 2016 and 2024, NIST conducted a multi-round standardization process for post-quantum primitives, culminating in three Federal Information Processing Standards published in August 2024 [3, 24–26]: FIPS 203 (ML-KEM, formerly CRYSTALS-Kyber) for key encapsulation [24]; FIPS 204 (ML-DSA, formerly CRYSTALS-Dilithium) for lattice-based signatures [25]; and FIPS 205 (SLH-DSA, formerly SPHINCS$^+$) for stateless hash-based signatures [26]. Earlier, NIST SP 800-208 [27] specified the stateful hash-based schemes XMSS and LMS. These primitives form the conformant signature surface that QGPL v3.0 §0 references; the algorithm-agility clause (QGPL §5.3) accommodates future NIST revisions, a property repeatedly identified as essential for any durable migration [28, 29].

The security assumptions differ usefully across schemes [30–33]. ML-DSA's security rests on Module Learning With Errors; SLH-DSA's rests only on preimage/collision resistance of the underlying hash—the most conservative assumption available, and the reason hash-based schemes

are repeatedly proposed for long-term and archival use [33, 34]. The size/performance tradeoff is correspondingly inverted, and we quantify it directly in §VIII-A: ML-DSA-65 produces ∼3.3 KiB signatures, while SLH-DSA-SHA2-256s produces ∼29 KiB signatures but a 64-byte public key. Implementation studies confirm both that ML-DSA is fast enough for constrained and high-throughput settings [35, 36] and that SLH-DSA's signing cost is its dominant practical drawback [37].

B. Quantum threat timeline and resource estimates

The urgency of the integrity problem is set by *when* a CRQC becomes available relative to the archival horizon of the data being protected. Resource-estimation work continues to refine the qubit and gate counts required to run Shor's algorithm against RSA and ECC; recent analyses sharpen the arithmetic of modular exponentiation and elliptic-curve point operations and, in some models, argue that a naive ion-trap realization is still far off [38, 39], while others stress that algorithmic and hardware progress has repeatedly compressed earlier estimates. Independent threat-timeline surveys place a non-negligible probability of a cryptographically relevant device within the 2030–2035 window [8, 40, 41]. The precise date is contested and, for our purposes, immaterial: what matters is that the date of first CRQC capability is plausibly *within* the lifetime of data released today, and that a signature, once forgeable, retroactively unprotects every artifact ever signed with the broken key. Confidentiality exposure (HNDL) and integrity exposure thus share a deadline, but integrity has the longer tail—a 2026 dataset re-examined in 2050 must still be attributable—which is precisely the regime the construction targets.

C. Existing distribution-integrity systems

Sigstore [14] introduced certificate-based "keyless" signing using short-lived X.509 certificates issued against OIDC identities and recorded in the Rekor transparency log. in-toto [15] formalizes multi-step supply-chain attestations; SLSA [16] provides a maturity framework for provenance; SBOMs provide a transparency substrate [17, 42]. A software-supply-chain systematization frames such systems through transparency, validity, and separation [18], and recent work extends artifact authentication to machine-learning datasets and models, where signing-and-verification overhead is a first-order concern [43]. All of these systems are classical in their default deployments (Ed25519, ECDSA-P256), and—critically for our argument—none is *required* by any license; they are toolchain conventions adopted at the producer's discretion.

D. Existing copyleft and open-data licenses

The GNU GPL v3 [19] introduced anti-Tivoization clauses and patent retaliation as license conditions, showing that conditions on distribution can be more than attribution requirements. The Creative Commons share-alike licenses [20] extend copyleft to non-software works; the Open Database License [21] addresses database rights. Empirical studies show that copyleft's viral condition is what drives both adoption decisions and compliance failures across package ecosystems [22, 23, 44]. None of these licenses addresses cryptographic distribution integrity.

E. The open-science archival horizon

Scientific datasets routinely outlive the cryptography that protects them at release. Genomic reference data published in 2005 is still cited; foundational astronomical catalogs span decades; particle-physics datasets from completed experiments remain authoritative. The FAIR principles and their governance make long-lived, reusable data an explicit community goal [10, 11, 45, 46], which only sharpens the integrity problem: if integrity is to be cryptographically verifiable across that horizon, the cryptography itself must outlive a CRQC. Mainstream threat models place the first cryptographically relevant device in the 2030–2040 window [8, 40]; resource-estimation work on Shor's algorithm against RSA/ECC continues to refine, but not eliminate, that horizon [38, 39]. Archived datasets need protection well beyond it.

III. Threat Model

We assume an adversary with:

- **Future CRQC access:** at some point within the archival horizon of the protected artifact (assumed conservatively ≥ 30 years), the adversary can execute Shor's algorithm [1] against any classical asymmetric scheme whose security depends on integer factorization or discrete logarithms—RSA, finite-field Diffie–Hellman, ECDH, DSA, ECDSA, EdDSA.
- **Read access to public distribution channels:** the adversary can observe published artifacts, manifests, and key documents. We do not assume confidentiality of distribution.
- **Capability to substitute artifacts under attacker-controlled channels:** the adversary can replace an artifact and its metadata in any distribution channel they compromise (a mirror, a CDN edge, a typosquatted package index).

We do **not** assume Grover-boosted hash collision capability beyond the quadratic speedup [13]; under that assumption SHA-256 retains ≈128-bit PQ security, sufficient when matched to NIST PQ Category 1 signature primitives. We do **not** assume compromise of the issuer's offline root key, though we discuss the consequences and mitigations in §V and §XIII.

Security goal. A QGPL-compliant distribution must be integrity-verifiable by a recipient at any point during its archival horizon, against an adversary capable of breaking any classical asymmetric primitive in use at the time of release. The construction must not require the recipient to trust a certificate authority, a transparency-log operator, or any centralized identity provider.

Non-goals. Confidentiality of the distribution; protection against denial of distribution; and protection against compromise of the artifact's *source* (a maliciously crafted dataset whose hash is honestly signed). These are out of scope: the construction protects integrity of distribution, not correctness of authorship or content.

A. The distribution-integrity game

We state the goal as a game between a challenger and an adversary $\mathcal{A}$ equipped with the capabilities above, including a CRQC oracle that breaks any classical asymmetric primitive. The challenger publishes a release (a, m)—artifact a and manifest m signed under an issuer key chain rooted at a fingerprint ϕ that the verifier has pinned out-of-band—together with a root-signed `keys.json`. $\mathcal{A}$ may observe all published bytes, query the CRQC oracle, and finally submit a pair $(a^\star, m^\star)$ with $a^\star \neq a$ to a verifier holding only ϕ. $\mathcal{A}$ *wins* if the verifier accepts $(a^\star, m^\star)$. The construction is *integrity-secure* if $\Pr[\mathcal{A}\text{ wins}]$ is negligible.

Proposition (informal). Under the assumptions that (i) SHA-256 is second-preimage resistant against a quadratic-speedup adversary, (ii) the chosen PQ signature scheme is EUF-CMA against a CRQC, and (iii) the issuer's offline root key is uncompromised, no $\mathcal{A}$ wins except with negligible probability. *Sketch.* Acceptance requires $\textsc{Sha256}(a^\star)$ to equal the hash field of an $m^\star$ that carries a valid PQ signature under a key chained to ϕ. With $a^\star \neq a$, either $m^\star$ reuses m's hash field—forcing a SHA-256 second preimage, excluded by (i)—or $m^\star$ carries a different hash, which requires a fresh valid PQ signature under a key the root vouches for. By (iii) $\mathcal{A}$ cannot mint a new `keys.json` entry, and by (ii) cannot forge under an existing signing key; the CRQC oracle helps only against the classical half of a hybrid, which the additive verifier treats as non-substitutive. Hence acceptance is negligible. The proposition's preconditions are exactly the residual risks catalogued in §IX: it says nothing when the root key is compromised, which is why that case is handled operationally rather than cryptographically.

IV. The Legal-Technical Construction

QGPL v3.0 grants the standard copyleft permissions (use, modify, distribute, publicly communicate) over a Work that may comprise Results (observation outputs—measurements, calculations, model outputs), datasets, documentation, and the Apparatus Code (software, scripts, notebooks, configurations) used to produce them. The novelty for the present argument lives in two clauses, whose composition is depicted in Fig. 1.

Section 2 (Copyleft). Modified Works must be distributed under QGPL v3.0 as a whole, with the License included, and downstream recipients must receive the same rights. Standard viral copyleft—but with the License's *other* conditions in scope.

Section 5 (Quantum-Safe Requirements). Distribution of the Work (or any Modified Work) requires **Quantum-Safe Distribution Metadata** for the distributed artifact: a cryptographic hash, a Quantum-Resistant signature over that hash (or over the artifact), the corresponding public key(s), and an unambiguous algorithm/parameter identifier. The metadata must be made available at the same level of accessibility as the artifact itself. Where cryptography is used for network-level access or transport, the distribution must support PQ key establishment (ML-KEM) or a hybrid construction; reliance on quantum-vulnerable public-key cryptography alone is non-compliant.

The combination is the construction. Because §2's copyleft requires downstream distributions to be QGPL, and because §5 requires PQ metadata on every distribution, a redistributor who fails to provide PQ metadata is in breach and their grant terminates (§7.1). The PQ requirement therefore propagates through the entire reuse chain of the Work, transitively, without any per-recipient agreement or central registry. This is not a "you should use PQ" recommendation; it is a *condition*: failure to comply terminates the legal permission to distribute. A redistributor who continues distribution without compliant metadata is in the same posture as one who removes the copyright notice or strips the license text—they have lost the grant. Table I positions this against neighboring integrity systems, none of which is bound to a license.

TABLE I
QGPL versus neighboring distribution-integrity approaches.

Property	Sigstore	in-toto/SLSA	PGP	**QGPL**
PQ signing (default)	no	no	no	**yes**
Mandated by license	no	no	no	**yes**
Propagates to derivatives	no	no	no	**yes**
No CA / log required	no	partial	yes	**yes**
Offline verifiable	partial	partial	yes	**yes**
Targets data & code	code	code	both	**both**

V. The Compliance Toolkit

A license condition is enforceable only to the extent that complying with it is feasible. The QGPL Compliance Toolkit Specification [47] (Draft v0.5.1) defines a portable manifest format and a reference CLI that make §5 compliance achievable today without external infrastructure.

A. Manifest format

A QGPL manifest is a JSON document canonicalized for signing per RFC 8785 (JCS) [48]. For a single artifact it carries: the artifact's hash (with algorithm); an optional SPDX expression for content license; author attribution (free-text name plus optional ORCID/DID); the signature (algorithm, base64 value, issuer-asserted `signed_at`); the signer's public key (algorithm, value, SHA-256 fingerprint); the issuer's name and a URI pointing to a `.well-known/qgpl/keys.json`; and an optional RFC 3161 timestamp for archival distributions. The signature is computed over the JCS canonicalization of the manifest

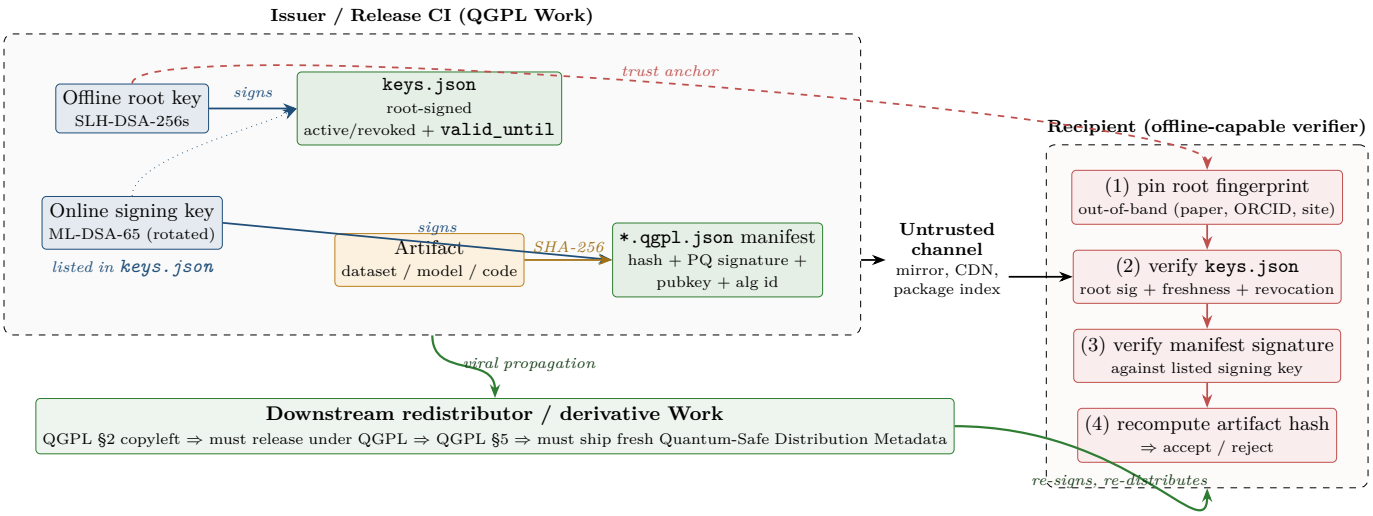


Fig. 1. QGPL architecture. An issuer keeps an *offline* SLH-DSA root key that signs a static `keys.json` enumerating *online* ML-DSA signing keys; each released artifact carries a JCS-canonicalized manifest with its hash, a PQ signature, the public key, and an unambiguous algorithm identifier. The recipient pins only the *root* fingerprint out-of-band and verifies the whole chain offline (steps 1–4); rotating a signing key never forces a re-pin. The copyleft clause (§2) and the quantum-safe clause (§5) compose: any downstream derivative must itself be QGPL and must therefore ship fresh quantum-safe metadata, so the PQ integrity floor propagates virally through the reuse graph.

with the signature value field set to the empty string, then the value is filled in—a standard self-referential signing pattern. This satisfies the four elements required by QGPL §0: (i) artifact hash, (ii) Quantum-Resistant signature, (iii) public key(s), and (iv) an unambiguous algorithm and parameter-set identifier.

B. Two-tier key hierarchy

The trust model uses no certificate authorities, no transparency logs, and no centralized identity providers (Fig. 1, left and right). The recipient pins a single root-key fingerprint out-of-band—from the project's website, an academic paper, a maintainer's ORCID page, or a `keys.json` document. The root key (recommended: SLH-DSA-SHA2-256s, offline, long-lived) signs a published `keys.json` that enumerates current and revoked signing keys (recommended: ML-DSA-65, online, rotated routinely). Day-to-day manifests are signed by signing keys; the root key never signs artifacts directly.

Crucially, the recipient pins the **root** key, not the signing key, so signing-key rotation never requires a re-pin. Revocation is an entry in `keys.json` carrying a timestamp and a reason (`rotation`, `compromise`, `superseded`, `lost`); subsequent verifications fail for any manifest signed by that key after `revoked_at`. A mandatory `valid_until` field (recommended $\leq$90 days) bounds the staleness of a cached snapshot that could otherwise indefinitely mask a revocation. This is the simplest construction we can identify that delivers meaningful rotation and revocation without infrastructure beyond a static HTTPS endpoint and the discipline to keep one key offline. The entire trust chain is two PQ signatures—one over the manifest, one over `keys.json`—both verifiable with no network access once `keys.json` is cached (we measure the cost in §VIII-C).

C. Composite hybrid signatures, additive only

The toolkit supports hybrid signatures combining a PQ algorithm with a classical one (e.g., `ML-DSA-65+Ed25519`), which the transition literature broadly recommends for the 2026–2030 window [4, 49–51]. The encoding is a structured `components` array, not a length-prefixed byte concatenation: each component carries its own algorithm name and base64 value, both components sign the *same* canonical manifest, and the top-level fingerprint is the SHA-256 of the JCS canonicalization of the components array. Component ordering must match the compound algorithm string (PQ first), making the canonical hash deterministic.

Verification is **strict**: every listed component must verify, and the toolkit deliberately provides no `policy`, `threshold`, or `required` field that would let a manifest declare "K of N components is sufficient." Hybrid in QGPL is an *additive* construction (defense in depth—the attacker must break every component) and never a *substitutive* one (threshold—the attacker need only break some subset). This is the inverse of the more permissive composite-signature semantics seen in some IETF drafts [52]; we argue the strict reading is correct for the integrity case, where graceful degradation under partial compromise is exactly what we want to refuse. A `--pq-only` mode exists for forensic and archival-recovery scenarios (e.g., a 2045 verifier examining a 2026 release whose Ed25519 key was later compromised); it skips classical components by algorithm-name match and requires at least one PQ component to verify. It is never the default; we evaluate it in §VIII-C.

TABLE II
WHY HYBRID MUST BE ADDITIVE. UNDER A STRICT (ADDITIVE) VERIFIER, AN ATTACKER MUST BREAK *EVERY* COMPONENT; UNDER A THRESHOLD VERIFIER, BREAKING THE WEAKEST SUFFICES. THE INTEGRITY CASE WANTS THE FORMER.

Future state (2045)	Additive (QGPL)	Threshold (rejected)
both halves intact	accept	accept
Ed25519 broken, ML-DSA ok	reject (default); accept w/ `--pq-only`	*accept silently*
ML-DSA broken, Ed25519 ok	reject (default)	*accept silently*
both broken	reject	reject

TABLE III
TRUST ASSUMPTIONS OF THREE INTEGRITY MODELS. QGPL TRADES AUTOMATIC IDENTITY BINDING FOR OFFLINE, INFRASTRUCTURE-FREE VERIFIABILITY.

Verify-time assumption	Sigstore	PGP	QGPL
Reachable CA / log	yes	no	no
Live identity provider	yes	no	no
Pinned trust anchor	no	yes	yes
Works fully offline	no	yes	yes
Automatic identity binding	yes	part.	no
PQ-secure by default	no	no	yes

VI. Design Rationale

Several of the toolkit's choices are deliberate departures from how supply-chain signing is usually done. We make the reasoning explicit, because the departures are what let the construction work without infrastructure.

A. JSON + JCS over ASN.1

The manifest is JSON canonicalized with RFC 8785 (JCS) rather than DER-encoded ASN.1. ASN.1 is more compact—a composite signature is ∼150 bytes smaller—but it is opaque to the inspection tools an open-science audience already has. A QGPL manifest can be read, diffed, and audited with `jq` or a text editor; its canonical form is reproducible by any JSON library that implements JCS; and a reviewer can confirm by eye that the signed bytes are the ones they think they are. For a construction whose entire value proposition is *verifiability by the recipient without trusting an intermediary*, the human-inspectability of the signed object is not a cosmetic preference but a security-relevant property. The wire-size penalty is paid once per artifact and, as §VIII-B shows, is dominated by the signature itself in every realistic case.

B. Additive, never substitutive hybrids

A hybrid signature can mean two opposite things. Under *additive* semantics the verifier requires every component to pass, so the attacker must break all of them—defense in depth. Under *substitutive* (threshold) semantics the verifier accepts if some subset passes, so the attacker need only break the weakest, making the hybrid *weaker* than its strongest component. Table II contrasts the two across the futures that matter. QGPL fixes the additive reading and provides no `threshold` field at all; graceful degradation under partial compromise, which is desirable for availability, is exactly the wrong default for integrity, where a silently accepted half-broken signature is indistinguishable from a forgery. The one case where tolerating a broken half is legitimate—an archivist in 2045 who knows Ed25519 has fallen and consciously wants the surviving PQ guarantee—is served by the opt-in `--pq-only` mode (§VIII-C), which makes the degradation a deliberate, logged operator decision rather than an automatic one.

TABLE IV
RECOMMENDED ALGORITHM BY ROLE, WITH THE NIST SECURITY CATEGORY AND THE PROPERTY THAT DRIVES THE CHOICE. MEASURED COSTS ARE IN TABLE V.

Role	Algorithm	Cat.	Driving property
Signing key	ML-DSA-65	3	best size/speed
High-assur. signing	ML-DSA-87	5	near-free upgrade
Offline root	SLH-DSA-256s	5	hash-only, 64 B key
Archival fast root	SLH-DSA-256f	5	faster ceremony
Transition hybrid	ML-DSA-65+Ed25519	3	+64 B depth
Hash	SHA-256	—	128-bit vs Grover

C. An offline root over Sigstore-style PKI

Sigstore binds identity through OIDC and records signatures in a transparency log; both require infrastructure the recipient must trust and reach. QGPL's two-tier model replaces that with a single out-of-band pin of a root fingerprint. The tradeoff is deliberate: we give up automatic identity binding (the toolkit does not establish *who* controls a key, only that a key signed an artifact) in exchange for the property that a recipient with a cached `keys.json` can verify a release in an air-gapped vault in 2050 with no live service in the loop. For archival science this is the right trade—the services that issue OIDC tokens and run transparency logs today are unlikely to be reachable, in their present form, across a thirty-year horizon, whereas a pinned hash in a printed paper is. The cost is that the issuer's identity rests on the out-of-band pin and the provenance statement (§XIII), not on a federated identity provider.

Table III makes the trade explicit: QGPL is closest to the PGP web-of-trust in requiring a pinned anchor and nothing online, but unlike PGP it is PQ-secure by default and unlike Sigstore it needs no reachable service. The deliberate sacrifice is automatic identity binding, recovered out-of-band by the provenance statement of §XIII.

D. Parameter and security-category selection

The toolkit pins defaults rather than leaving primitive choice to the adopter, because the failure mode of "pick your own" is a long tail of weak or mismatched selections—the same crypto-agility-without-guidance problem the migration literature documents [9, 28]. Table IV summarizes the recommendations and the property driving each. The guiding rule is security-level matching: a 128-bit-PQ hash (SHA-256, adequate against Grover's quadratic speedup) paired with a Category-3-or-higher signature, and a Category-5 hash-based root whose security rests on the most conservative assumption available. Because the three ML-DSA parameter sets differ by under 1.3× in latency (§VIII-A), choosing a higher category on the signing path is nearly free, so the default errs toward more security rather than less.

E. Timestamping and transparency, deferred deliberately

A third-party RFC 3161 timestamp would let a verifier establish that a signature existed before a given moment—useful for the "signed before the key was broken" argument. The toolkit reserves a structured `timestamp` slot but does *not* require a token in v1, because RFC 3161 timestamp authorities with PQ-secure signatures are not yet deployed; mandating a token whose own cryptography is classical would reintroduce the very weakness the construction removes. The slot is structured (not a bare string) so that a PQ-signed transparency-log inclusion proof can be added in v1.1 without a breaking change. This is the same "recommend, don't require, until the dependency is itself quantum-safe" discipline that the transition literature urges for every link in the chain [5, 53].

VII. Reference Implementation and Experimental Methodology

To move from assertion to measurement we built `qgpl_ref`, a spec-faithful reference of the toolkit's signing/verification core. It implements the manifest format and self-referential JCS signing rule, the two-tier `keys.json` hierarchy, and the additive hybrid encoding exactly as specified. PQ primitives are provided by `quantcrypt` [54], which wraps the PQClean reference implementations of the final NIST parameter sets; classical primitives (Ed25519, ECDSA-P256, RSA-3072) come from the `cryptography` library; JCS canonicalization follows RFC 8785. We verified that key and signature sizes match the FIPS 204/205 parameter sets exactly (e.g., ML-DSA-65 public key 1952 B, signature 3309 B; SLH-DSA-SHA2-256s public key 64 B, signature 29 792 B), confirming the backend exercises real standardized primitives rather than stand-ins.

Platform. All measurements were taken on a single Apple-silicon machine (`arm64`, 10 cores, macOS/Darwin 25.5), Python 3.13, single-threaded. Latencies are wall-clock per-operation times from an adaptive micro-benchmark: two warmup iterations, then repetitions until a 2 s budget or a cap is reached (up to 300 repetitions for sub-millisecond operations, fewer for SLH-DSA signing). We report the median and population standard deviation; the signed message in the latency benchmarks is the JCS canonicalization of a real single-artifact manifest, so signing cost reflects the actual signed object. The harness and raw `results.json` are in the repository, and every figure and table number below is regenerated from that file.

Scenarios. We organize the evaluation around three usage scenarios that exercise different parts of the construction: *(A)* a publisher choosing a signing primitive (§VIII-A); *(B)* an institution releasing artifacts at scale (§VIII-B); and *(C)* a downstream—or future archival—recipient verifying, rejecting tampered releases, and recovering a release whose classical half is later broken (§VIII-C).

Algorithm 1: SignRelease(artifact a, signing key k)

1) $h \leftarrow \textsc{Sha256}(a)$
2) build manifest m with m.artifact.hash $\leftarrow h$, public key, algorithm id; set m.signature.value $\leftarrow \varepsilon$
3) $c \leftarrow \textsc{JcsCanonicalize}(m)$
4) $\sigma \leftarrow \textsc{Sign}_k(c)$
5) m.signature.value $\leftarrow \textsc{Base64}(\sigma)$; emit m, detached σ

(hybrid) blank *all* component values, canonicalize once, each component key signs the same c, fill each component value.

Fig. 2. Self-referential JCS signing. The signature covers the canonical manifest with its own value field blanked; in the hybrid case every component signs the identical canonical bytes, so each is independently verifiable.

Algorithm 2: VerifyRelease(manifest m, artifact a, pinned root fp)

1) **if** $\textsc{Sha256}(a) \neq m$.artifact.hash **return** REJECT(hash)
2) fetch/cached `keys.json`; **if** root fp $\neq$ pinned **return** REJECT
3) **if** `keys.json` stale or root sig invalid **return** REJECT
4) **if** signing key revoked before m.signed_at **return** REJECT
5) $c \leftarrow \textsc{JcsCanonicalize}(m$ with signature blanked$)$
6) **for each** component i: **if** $\neg\textsc{Verify}_{pk_i}(c, \sigma_i)$ **return** REJECT
7) **return** ACCEPT (`--pq-only`: skip classical i, require ≥ 1 PQ pass)

Fig. 3. Strict verification. Every component must pass; the optional `--pq-only` path is the only way a classical-component failure is tolerated, and only by explicit operator choice.

VIII. Evaluation

A. Scenario A: per-release cryptographic cost

The cost a publisher pays per release is dominated by one signing operation and the metadata it produces; a recipient pays one verification. Table V and Fig. 4 report keygen, sign, and verify latency and artifact sizes across the conformant signature surface, with classical baselines and two hybrids.

The default signing primitive, ML-DSA-65, signs a release manifest in a median 0.72 ms and verifies it in 0.37 ms—i.e., a publisher's marginal signing cost and a recipient's marginal verification cost are both sub-millisecond, and neither scales with the number of recipients. The three ML-DSA parameter sets differ by less than

TABLE V
SCENARIO A: COST OF THE QGPL CONFORMANT SIGNATURE SURFACE AND CLASSICAL BASELINES. SIZES IN BYTES; LATENCIES ARE MEDIANS IN MILLISECONDS ON APPLE-SILICON `arm64`, SINGLE-THREADED (§VII). PQ, CLASSICAL, HYBRID.

Algorithm	Pub key (B)	Sig (B)	Keygen (ms)	Sign (ms)	Verify (ms)	Role in QGPL
ML-DSA-44	1312	2420	0.49	0.65	0.35	lightweight signing
ML-DSA-65	1952	3309	0.42	0.72	0.37	**default signing key**
ML-DSA-87	2592	4627	0.47	0.79	0.43	high-security signing
SLH-DSA-SHA2-256s	64	29792	75.0	889.2	1.50	**offline root / archival**
SLH-DSA-SHA2-256f	64	49856	4.98	94.5	2.79	fast-signing archival variant
Ed25519	32	64	0.09	0.09	0.20	classical hybrid half
ECDSA-P256	65	70	0.02	0.03	0.08	classical hybrid half
RSA-3072	422	384	236.3	2.61	0.11	legacy interop baseline
ML-DSA-65+Ed25519	1984	3373	—	0.91	0.69	**recommended transition hybrid**
ML-DSA-65+ECDSA-P256	2017	3379	—	0.90	0.57	transition hybrid (ECDSA)

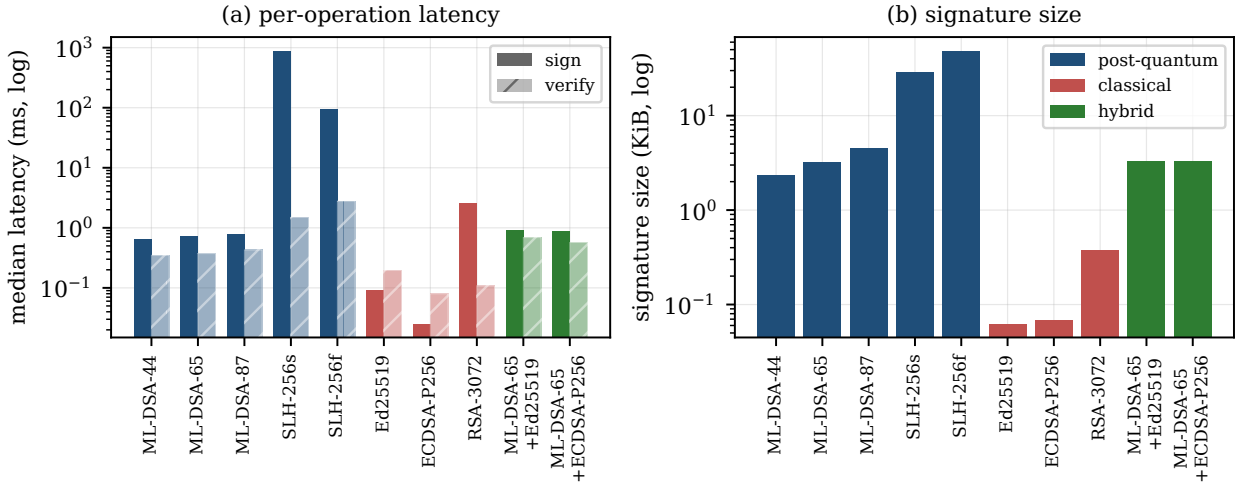


Fig. 4. Scenario A. (a) Per-operation median latency (log scale): ML-DSA signs and verifies in well under a millisecond; SLH-DSA-256s signing is three orders of magnitude slower (∼0.89 s) but is used only for the offline root, which signs `keys.json` a handful of times per year. (b) Signature size (log scale): the hybrids (green) sit essentially on top of ML-DSA-65 because the added Ed25519/ECDSA half is only ∼64–70 bytes.

a factor of 1.3 in latency, so choosing a higher security category is nearly free on the signing path. SLH-DSA tells the opposite story: SLH-DSA-256s signing is ∼0.89 s—three orders of magnitude slower than ML-DSA—which is exactly why the construction confines it to the *offline root*, which signs only the short `keys.json` document a few times per year, and never signs artifacts. Its 64-byte public key and conservative hash-only security assumption are ideal for the long-lived pin, and its 29 KiB signature is paid once per `keys.json`, not once per release.

The most consequential result for the transition window is the cost of going hybrid. Adding an Ed25519 half to ML-DSA-65 raises the signature from 3309 to 3373 bytes (+64 B, +1.9%) and the sign+verify path by under 0.3 ms combined, while the public key grows by 32 bytes. Defense-in-depth against "the lattice assumption was wrong" is therefore available at negligible marginal cost, which supports the toolkit's recommendation to ship hybrids through 2030 and matches the "compact hybrid" findings of the transition literature [49, 50]. For reference, RSA-3072 keygen alone (236 ms) is more expensive than a full ML-DSA-65 sign+verify cycle by two orders of magnitude, a reminder that "classical" is not a synonym for "cheap."

The transport profile (ML-KEM). QGPL §5.2 additionally requires PQ key establishment wherever cryptography protects network access or transport. Although the toolkit targets artifacts-on-disk and leaves the wire to PQ-TLS stacks [53, 55], we measured the FIPS 203 ML-KEM primitives on the same platform for completeness (Table VI). Key generation, encapsulation, and decapsulation are all sub-millisecond across the three parameter sets, and ciphertexts are 0.75–1.5 KiB—confirming that the transport half of §5 is, like the signing half, cheap enough that the binding constraint is deployment and configuration, not cryptographic cost [49].

B. Scenario B: metadata overhead at release scale

The central objection to a *mandatory* metadata condition is that it taxes every release. We quantify that tax in two regimes: a single artifact of varying size, and

TABLE VI
TRANSPORT PROFILE: FIPS 203 ML-KEM ON THE EVALUATION PLATFORM. SIZES IN BYTES (SHARED SECRET IS 32 B FOR ALL); LATENCIES ARE MEDIANS IN MS.

KEM	Pub key	Ciphertext	Keygen	Encaps	Decaps
ML-KEM-512	800	768	0.32	0.47	0.38
ML-KEM-768	1184	1088	0.32	0.48	0.39
ML-KEM-1024	1568	1568	0.34	0.49	0.40

TABLE VII
THE POST-QUANTUM "METADATA PREMIUM." SIGNATURE/KEY SIZES ARE MEASURED (TABLE V); METADATA FOR ML-DSA-65 AND SLH-DSA-256S IS MEASURED, THE CLASSICAL AND HYBRID ROWS ARE DERIVED FROM EQ. (1). THE LAST COLUMN IS RELATIVE OVERHEAD ON A 1 MIB ARTIFACT.

Signing algorithm	Sig (B)	Metadata (KiB)	@ 1 MiB
Ed25519 (classical)	64	0.94	0.09%
ECDSA-P256 (classical)	70	1.0	0.10%
RSA-3072 (classical)	384	2.2	0.21%
ML-DSA-65 (PQ default)	3309	10.8	1.06%
ML-DSA-65+Ed25519	3373	11.6	1.10%
SLH-DSA-256s (archival)	29792	68.7	6.7%

TABLE VIII
SCENARIO C1: CONFORMANCE DETECTION MATRIX. EACH CASE IS THE CORRESPONDING NEGATIVE CASE OF THE REPOSITORY FIXTURE; "CORRECT" MEANS THE VERIFIER'S VERDICT MATCHED THE SPEC-MANDATED OUTCOME.

Case	Expected	Verdict	Correct
positive (well-formed release)	accept	accept	✓
hash mismatch (artifact mutated)	reject	reject	✓
bad signature (one byte flipped)	reject	reject	✓
revoked key (after `revoked_at`)	reject	reject	✓
stale `keys.json` (past `valid_until`)	reject	reject	✓
hybrid, classical half broken (strict)	reject	reject	✓

an aggregate `MANIFEST` covering many artifacts under one signature (Fig. 5).

Single artifact. A per-artifact ML-DSA-65 manifest plus its detached signature is ∼11.1 KiB regardless of artifact size—the signature, public key, and JSON envelope dominate, and the hash is a fixed 32 bytes. Relative overhead therefore decays as the inverse of artifact size: 6127% for a 181-byte fixture, 108% at 10 KiB, 1.06% at 1 MiB, and 106 parts-per-million at 100 MiB. The crossover below 1% of payload occurs at roughly 11 KiB—smaller than essentially any real research-data artifact. With the archival SLH-DSA-256s primitive the fixed cost is larger (∼70.4 KiB) but the same 1/size decay applies, crossing 1% near 70 KiB. In other words, the mandatory-metadata tax is regressive only for artifacts too small to matter; for the megabyte-to-gigabyte datasets that dominate open science it is in the parts-per-million range. End-to-end signing of a 100 MiB artifact takes 47 ms, of which hashing—not the PQ signature—is the dominant term, so the cost tracks I/O, not cryptography.

Aggregate manifest. Real releases bundle many files (a dataset, the code that produced it, a paper, a README). A single aggregate `MANIFEST.qgpl.json` carries one signature over an `artifacts` array, so the expensive per-signature cost is paid once. Per-artifact amortized metadata falls from 7.6 KiB at N=1 to 886 B at N=10 and 148 B at N=1000, while total signing time rises only from 1.1 ms to 6.4 ms across that three-order-of-magnitude range. A thousand-file release is thus signed in single-digit milliseconds and carries ∼145 KiB of total integrity metadata—a fraction of a percent for any realistic payload. The construction therefore scales the way an institutional release pipeline needs it to: bounded, predictable, and dominated by hashing rather than by signing.

An overhead model. The measurements fit a simple closed form. A single-artifact manifest's size is essentially independent of the artifact and is

$$M \approx E + \tfrac{7}{3}\, s_{\text{sig}} + \tfrac{4}{3}\, s_{\text{pk}}, \qquad (1)$$

where $E \approx 765$ B is the fixed JSON envelope, s_{sig} and s_{pk} are the raw signature and public-key sizes, the factor $\frac{4}{3}$ is base64 expansion, and the extra 1 on the signature accounts for the raw detached `.qgpl.sig`. Equation (1) predicts 11.1 KiB for ML-DSA-65 and 70.4 KiB for SLH-DSA-256s, matching the measured values to within the envelope's variation. Relative overhead is then $\rho(n) = M/n = \Theta(1/n)$, so the sub-1% crossover sits at $n > 100M$ (∼11 KiB for ML-DSA-65), and for an aggregate of N artifacts the shared signature gives per-artifact overhead $E'/N + h$ where $h \approx 140$ B is one hash entry—hence the 148 B asymptote at N=1000.

The cost of going post-quantum is therefore a roughly 11× increase in *fixed* metadata over a classical Ed25519 baseline (Table VII)—real, but a one-time per-artifact cost that is still ∼1% of a one-megabyte payload and vanishes for larger data. The hybrid premium over PQ-alone is under 1 KiB. This is the quantitative core of our claim that mandating PQ metadata is affordable: the premium is bounded, paid in bytes rather than latency, and shrinks against exactly the large-artifact releases for which long-horizon integrity matters most.

C. Scenario C: verification, conformance, and archival recovery

The recipient's path is where a distribution-integrity construction earns its keep. We evaluate three things: that the verifier accepts a good release and rejects every defined failure mode; what the full two-signature trust chain costs; and whether a release survives the eventual break of its classical half.

Detection. Table VIII and Fig. 6(a) show the verifier reproducing the spec-mandated verdict on all six conformance cases: it accepts a well-formed release and rejects a mutated artifact (hash mismatch), a single-byte signature corruption, a manifest signed by a revoked key after its `revoked_at` time, a `keys.json` past its `valid_until`, and—under the default strict policy—a hybrid whose classical component fails. The last case is the additive property in action: a broken half is fatal, never tolerated.

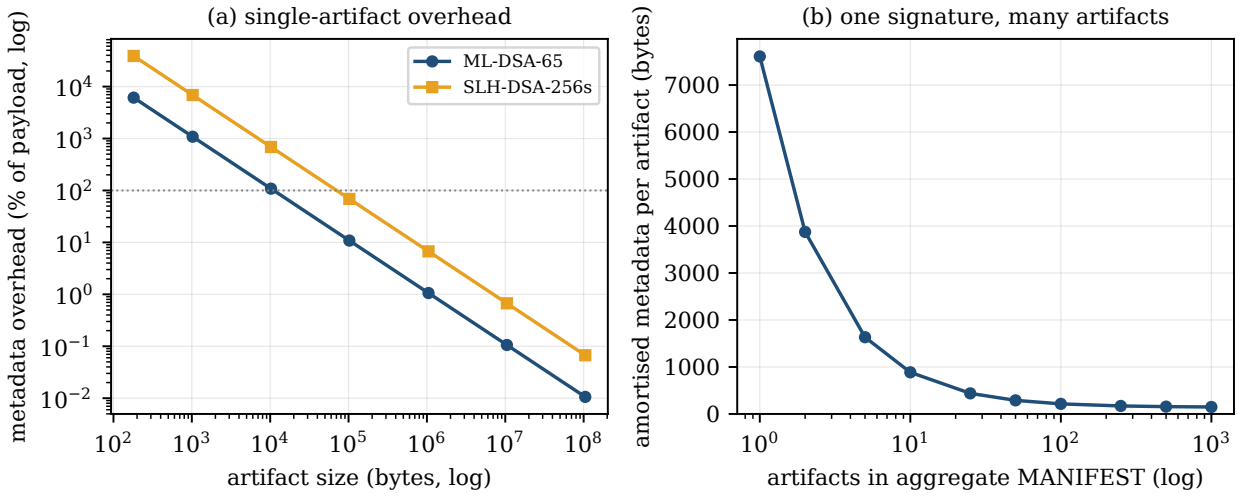


Fig. 5. Scenario B. (a) Relative metadata overhead versus artifact size (log–log). QGPL metadata is essentially fixed in size (∼11.1 KiB for ML-DSA-65, ∼70.4 KiB for SLH-DSA-256s), so relative overhead falls as 1/size and crosses below 1% of payload at ∼11 KiB (ML-DSA-65). (b) Amortized metadata per artifact in a single aggregate `MANIFEST`: one signature covers N artifacts, so per-artifact overhead falls from 7.6 KiB at N=1 to 148 bytes at N=1000.

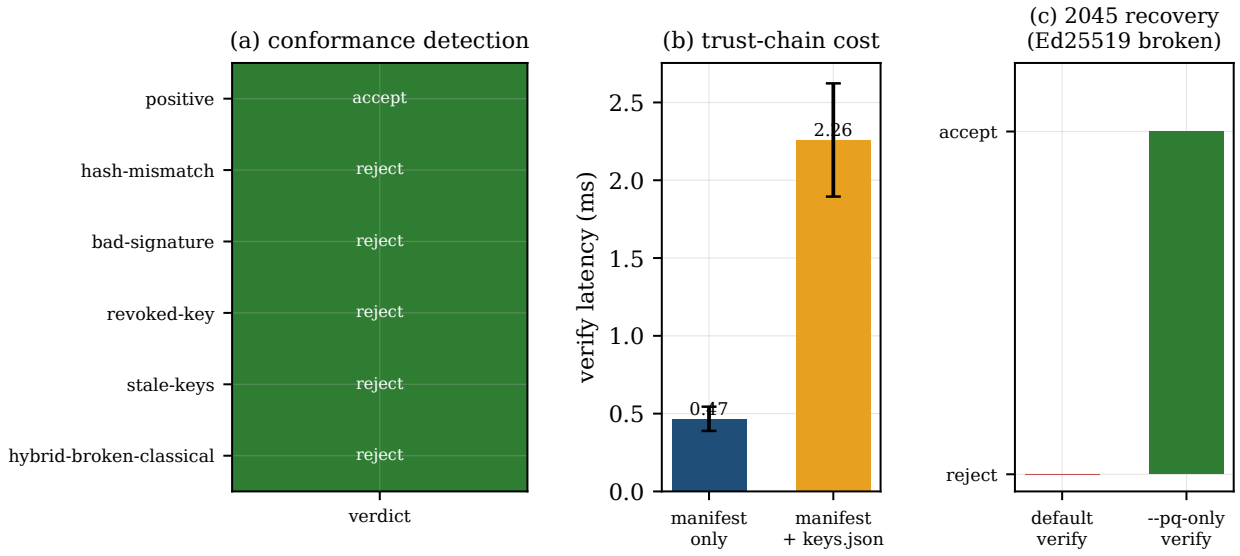


Fig. 6. Scenario C. (a) All six conformance cases produce the spec-mandated verdict (6/6 correct). (b) Trust-chain verification cost: a manifest-only check is 0.47 ms; adding the `keys.json` root check (one SLH-DSA-256s verification) brings the full offline chain to 2.26 ms. (c) Archival recovery: a hybrid release whose Ed25519 half is broken in 2045 is *rejected* by the default strict verifier and *recovered* by the opt-in `--pq-only` mode, which checks the surviving ML-DSA half.

Trust-chain cost. Verifying the manifest signature alone is a median 0.47 ms. Verifying the full offline chain—manifest signature plus the SLH-DSA-256s signature over `keys.json`—is 2.26 ms (Fig. 6(b)). The increment is the single SLH-DSA verification, consistent with the 1.50 ms SLH-DSA-256s verify cost in Table V; it is paid once per cached `keys.json`, not once per artifact, so a recipient verifying a thousand-artifact release pays it once. Both numbers are network-free: once `keys.json` is cached, the recipient needs no CA, no log, and no connectivity.

Archival recovery. Finally we exercise the scenario the hybrid mode exists for. We take a `ML-DSA-65+Ed25519` release and simulate a 2045 world in which Ed25519 has fallen, by corrupting the classical component. The default strict verifier rejects it (verdict `component-fail:Ed25519`) because additive semantics require both halves—correct behavior at release time. The opt-in `--pq-only` verifier, intended precisely for forensic and archival recovery, skips the broken classical component, checks the surviving ML-DSA half, and accepts (Fig. 6(c)). The release thus degrades gracefully *by explicit operator choice* rather than silently: an archivist in 2045 can still establish PQ-backed integrity of a 2026 artifact, while a 2026 recipient is never quietly handed a half-broken signature. This is the

operational payoff of designing hybrids to be additive rather than threshold.

Throughput. The sub-millisecond verification cost has a practical corollary for repositories that verify at ingest. At the measured 0.47 ms manifest-verification median, a single core checks ∼2130 manifests per second; even the full two-signature chain at 2.26 ms sustains ∼440 releases per second per core, and the chain's extra cost is paid once per cached `keys.json` rather than per artifact. A repository ingesting tens of thousands of QGPL deposits per day therefore spends seconds of CPU per day on integrity verification—negligible against the I/O of storing the data itself. PQ verification is not the bottleneck anyone feared; signing-side ceremony discipline (§XIII) is the real operational cost.

D. Reproducibility and measurement variance

Two properties make the numbers above stable. First, every *size* is deterministic: signature and key lengths are fixed by the parameter set, base64 expansion is exact, and the JSON envelope varies only with field contents, so Tables V, VII, and VI are reproducible to the byte on any conformant implementation. Second, the *latencies* are low-variance on the signing and verification paths that matter: the manifest verification median is 0.47 ms with a population standard deviation of 0.078 ms (∼17%), and ML-DSA-65 signing sits near 0.72 ms with comparable spread; the SLH-DSA root operation is intrinsically slower but is off the per-release path. We report medians precisely because the rare scheduling outlier inflates the mean without reflecting steady-state cost. The harness fixes the signed message (a real canonical manifest), warms the cache before measuring, and budgets each operation independently, so a re-run on the same class of hardware reproduces the medians within a few percent. The committed `results.json` and `make_figures.py` regenerate every figure and table from a single command, so a reviewer can reproduce the entire evaluation, not merely inspect it.

E. Threats to validity

Our latencies are single-machine, single-threaded, and specific to PQClean reference code on Apple silicon; absolute numbers will differ on other microarchitectures and with vectorized or hardware implementations, which the literature shows can move ML-DSA and SLH-DSA costs substantially [35, 36]. The *relative* structure of our results—ML-DSA sub-millisecond, SLH-DSA signing dominated by the root ceremony, hybrids near-free, metadata overhead $\Theta(1/\text{size})$—is a property of the construction and the standardized parameter sizes, not of the platform, and should carry across environments. The reference exercises the signing/verification core, not the full CLI (`audit`, network `keys.json` fetch, RFC 3161 timestamping); those paths are I/O- or policy-bound and out of scope for cryptographic cost. We make no claim about the *legal* efficacy of the construction; §XIII treats that honestly.

TABLE IX
THREAT-TO-DEFENSE MAPPING. "RESIDUAL" MARKS THE GAP THE DESIGN DOES NOT CLOSE CRYPTOGRAPHICALLY.

Adversary capability	QGPL defense
Forge classical signature with CRQC	PQ signature is the load-bearing one; classical is only an additive bonus
Substitute artifact on a mirror/CDN	hash binding in a PQ-signed manifest; substitution fails verification
Compromise an online signing key	revoke in `keys.json`; rotate under unchanged root pin
Replay an old `keys.json` to hide a revocation	mandatory `valid_until` bounds cache staleness
Downgrade to a weak/-classical alg	explicit alg id is signed; verifier policy rejects non-conformant algs
Break one hybrid half in the future	strict additive verify rejects; `--pq-only` recovers by choice
Compromise the offline root key	*residual:* catastrophic; mitigated by OPSEC + disclosure, not proof
Fabricate the "offline" provenance	*residual:* no HW attestation; multi-location pin + disclosure only

IX. Security Analysis

We now argue, threat by threat, that the construction meets the security goal of §III—integrity-verifiability across the archival horizon against an adversary who can break any classical asymmetric primitive—and we are explicit about the boundaries of that claim. The analysis maps each adversary capability to the design element that defends against it (summarized in Table IX).

A. Artifact substitution and forgery

The core threat is substitution: the adversary replaces a published artifact and its metadata on a channel they control. Verification recomputes the artifact's SHA-256 and compares it to the hash in the manifest, and the manifest is bound to the issuer's key by a PQ signature over its JCS canonicalization. To pass verification the adversary must therefore either find a SHA-256 second preimage (infeasible even with Grover's quadratic speedup at 256 bits [13]) or forge an ML-DSA/SLH-DSA signature (the assumption we explicitly place out of the adversary's reach, since these are the primitives chosen to survive a CRQC [25, 26, 33]). A classical signature on the same artifact provides no help to the adversary and no obligation on the verifier: in the additive hybrid it is an extra hurdle, and in the PQ-only manifest it is absent. This is exactly the property that mainstream classical-only systems [14–16] lack: against a CRQC their single signature is forgeable, and substitution succeeds. Scenario C1 (Table VIII) demonstrates detection of both the hash-mismatch and signature-corruption cases.

B. Signing-key compromise and rotation

Online signing keys live on CI infrastructure and are the most plausible compromise. The two-tier hierarchy makes this recoverable: the issuer publishes a `revoked_keys` entry (reason `compromise`) in a freshly root-signed `keys.json`, and

verifiers reject any manifest signed by that key after its `revoked_at` timestamp. Because the recipient pins the *root* fingerprint, rotation and revocation require no re-pin—the property that makes routine key hygiene operationally viable, in contrast with a flat single-key model whose only recovery is to re-establish the project's identity from scratch [56]. Scenario C1's revoked-key case confirms the verifier rejects a manifest from a revoked key.

C. Rollback and stale-cache attacks

An adversary who has had a key revoked benefits from convincing a verifier to keep using an *old* `keys.json` that predates the revocation. The mandatory `valid_until` field bounds this: a cached document past its validity triggers a warning and, past a grace window, a hard refusal, so a captured cache cannot indefinitely mask a revocation. This is a deliberately weaker guarantee than a transparency log's append-only freshness, traded for the no-infrastructure property; the residual exposure is bounded by the $\leq$ 90-day `valid_until` window rather than eliminated. Scenario C1's stale-`keys.json` case exercises this path.

D. Downgrade and algorithm substitution

Because every signature, key, and hash carries an explicit, *signed* algorithm identifier, an adversary cannot silently present a weak-algorithm signature as a strong one—the identifier is inside the signed bytes, so altering it invalidates the signature. A verifier policy can additionally refuse non-conformant algorithms outright (e.g., reject a manifest whose `signature.alg` is a bare classical scheme), enforcing §5 at verification time. The component-ordering invariant in the hybrid encoding (PQ first) closes a related ambiguity: two component orderings cannot yield the same fingerprint, so a reordering attack cannot produce a colliding key identity.

E. Partial compromise of a hybrid

The additive semantics analyzed in §VI and Table II are a security property, not a convenience: a future in which one of the two algorithms falls must not silently downgrade to acceptance. The strict verifier rejects a hybrid with any failing component, and the only path to accepting a partially-broken release is the explicit `--pq-only` mode, which (i) is never the default, (ii) requires at least one PQ component to verify, and (iii) refuses outright if all components are classical. Scenario C3 demonstrates both halves of this: default rejection of the broken hybrid, and deliberate PQ-only recovery.

F. Residual risks and limits of the claim

Two gaps are not closed cryptographically and we do not claim otherwise. *Root-key compromise* is catastrophic—an adversary who holds the offline root can mint a `keys.json` that any pinning recipient will trust—and is addressed only by operational discipline (air-gapped generation, split custody, rare ceremonies) and the bounded blast radius of out-of-band re-pinning. *The provenance claim* that the root was generated offline is, as §XIII details, unattestable in v1; the mitigation is multi-location pinning plus a specific written disclosure, which raises the cost of fabrication without eliminating it. Both gaps are inherent to a no-infrastructure trust model and are the explicit price of not depending on a CA or a transparency log; closing them requires hardware attestation, which is on the roadmap and blocked on HSM vendor support for PQ keygen. Within those stated boundaries, the construction meets its goal: a recipient at any point in the archival horizon, holding only an out-of-band root pin, can establish PQ-backed integrity of a release without trusting any third party.

X. Conformance Fixture

To make the construction implementable and reviewable before a full reference toolkit ships, the repository carries a conformance fixture pinned to the current spec draft [57]: a 181-byte CSV artifact with a real, stable SHA-256 baked into a template manifest, a placeholder-signed `keys.json`, a worked operational-provenance statement (§XIII), and a reference generator that produces fully-signed positive and negative cases—hash mismatch, signature corruption, revoked key, stale `keys.json`, and a hybrid with a broken classical component. The fixture's README enumerates expected outcomes per case, mapped to the spec section each exercises; these are exactly the cases reproduced in Table VIII. The committed form keeps clearly-marked placeholders for every cryptographic field, with real bytes produced only when an implementer runs the generator. This is the artifact reviewers should run a candidate implementation against to claim compliance, and the artifact we use to bootstrap the toolkit: its first production release will be PQ-signed using the generator described here.

XI. A Worked QGPL Release

To make the construction concrete, we walk through the artifacts a solo researcher produces when releasing a single dataset under QGPL. The release directory holds the artifact, its manifest and detached signature, the license text, and a `NOTICE`; the issuer additionally publishes a `keys.json` and a one-paragraph provenance statement at a stable URL. Listing 1 shows the signed manifest (base64 values elided): it carries the artifact hash, the ML-DSA-65 signature over the canonical bytes, the signing public key with its fingerprint, and an `issuer.uri` pointing at the key document. Listing 2 shows the root-signed `keys.json`: the offline SLH-DSA-256s root key, the active ML-DSA-65 signing key, an (empty) revocation list, and the mandatory `valid_until`.

Provenance statement. Because the toolkit cannot attest that the root key is genuinely offline (§XIII), the issuer publishes specifics alongside the pinned fingerprint—for example: *"Root key generated 2026-02-15 on a Tails 6.x USB boot of a single-use laptop with no network interface*

Listing 1. Single-artifact manifest `results-2026-06-13.csv.qgpl.json` (base64 elided). The signature covers the JCS canonicalization of this object with `signature.value` blanked.

```
{
  "qgpl_spec": "1.0",
  "license": "QGPL-3.0",
  "artifact": {
    "name": "results-2026-06-13.csv",
    "size": 181,
    "hash": { "alg": "SHA-256", "value": "d8971010...c04a7" },
    "license_spdx": "CC-BY-4.0"
  },
  "author": { "name": "Dr. Sarunas Grigaliunas", "orcid": "0000-0001-9268-9244" },
  "signature": { "alg": "ML-DSA-65", "value": "kQ4z...A==", "signed_at": "2026-06-13T12:00:00Z" },
  "public_key": { "alg": "ML-DSA-65", "value": "Hh1c...Q==", "fingerprint": "sha256:1a2b3c..." },
  "issuer": { "name": "The Open Reality Initiative -- Release CI",
              "uri": "https://qgpl.org/.well-known/qgpl/keys.json" }
}
```

Listing 2. Root-signed `keys.json`. The recipient pins `root_key.fingerprint` once; signing keys rotate beneath it without a re-pin. `valid_until` bounds how long a cached copy can mask a revocation.

```
{
  "qgpl_keys_spec": "1.0",
  "issuer": "The Open Reality Initiative",
  "issued_at":   "2026-06-13T12:00:00Z",
  "valid_until": "2026-09-11T12:00:00Z",
  "root_key":    { "alg": "SLH-DSA-SHA2-256s", "value": "T0g...", "fingerprint": "sha256:rootfp..." },
  "signing_keys": [
    { "fingerprint": "sha256:sk1fp...", "alg": "ML-DSA-65", "value": "Hh1c...Q==",
      "valid_from": "2026-01-01T00:00:00Z", "valid_until": "2026-12-31T23:59:59Z",
      "purpose": "release-ci" }
  ],
  "revoked_keys": [],
  "signature": { "alg": "SLH-DSA-SHA2-256s", "value": "QmtZ...", "signed_at": "2026-06-13T12:00:00Z" }
}
```

enabled; private key split across two YubiHSM 2 devices held by the maintainer and one trustee in separate cities; signing ceremonies require both and are logged." This is not cryptographic proof; it is the substrate an independent reader uses to judge the claim, and the thing the issuer can be held to if a divergence later surfaces. Pinning the same fingerprint in several independent places—the project site, this paper, the maintainer's ORCID page—protects the statement itself from being silently swapped.

What a recipient does. A recipient runs `VerifyRelease` (Algorithm 2): recompute the artifact hash, fetch or use a cached `keys.json`, check its root signature and freshness against the pinned fingerprint, confirm the signing key is not revoked, and verify the manifest signature over the canonical bytes. As §VIII-C measures, the whole chain is ∼2.3 ms and needs no network once `keys.json` is cached.

XII. Deployment and Integration

QGPL metadata is designed to drop into workflows researchers already use. *Archives:* manifests, signatures, and `keys.json` are ordinary files, so they upload unchanged into Zenodo, Figshare, and DOI-archive deposits alongside the data they cover, giving every deposit a PQ-verifiable integrity layer without archive-side changes. *CI:* a release pipeline pulls the signing key from a secret store, signs a configured glob, and uploads the `*.qgpl.json`/`*.qgpl.sig` pair as release assets; §VIII-B shows a thousand-artifact aggregate signs in single-digit milliseconds, so the step is negligible in a build. The expensive, rare operation—signing a new `keys.json` with the offline root—happens on a quarterly cadence by sneakernet, off the CI runner entirely. *Co-existence with SLSA and Sigstore:* QGPL is orthogonal to provenance frameworks. A release can be QGPL *and* SLSA Level 3; the toolkit can read a Sigstore bundle and *augment* it with a PQ signature so both ship together, but a classical Sigstore signature alone does not satisfy §5—the point of the construction is precisely that the mandated signature is post-quantum. *Slim verifier:* a recipient who only verifies needs neither a CA nor a heavyweight PQ stack; a verifier carrying just a pure-software ML-DSA implementation and a JCS library can check a release offline, which is what makes archival verification in 2050 plausible.

XIII. Discussion and Limitations

A. OSI / FSF approval

QGPL is, by design, a copyleft license whose conditions extend beyond pure copyright. Section 5's mandatory PQ requirement is a *technical condition on distribution*, not a restriction on the field of use of the Work, but the distinction may be litigated. The SSPL precedent suggests that licenses imposing operational conditions on redistributors face significant pushback in the OSI approval process. We do not yet know whether QGPL will clear OSI/FSF review; the strategic choice is whether to seek approval (and potentially soften §5 to a strong recommendation) or to position QGPL as an opinionated non-OSI license aimed at scientific-data and government/EU contexts where the stricter posture is an asset. Both paths are defensible; we defer the choice to the project's governance.

B. Conditions over uncopyrightable factual data

A Result released under QGPL may be a pure measurement—a number, a tuple, a fact. In many jurisdictions facts are not copyrightable, and the copyleft hook over a pure factual Result may have no teeth. QGPL §1.3 addresses this by framing the License as a "public permission and covenant not to sue" under whatever rights the Licensor may hold (including sui generis database rights). This is defensible but weakens the construction precisely where the open-science case is strongest. We note that the Apparatus Code accompanying a release is unambiguously copyrightable, and that the manifest, signatures, and `keys.json` are themselves copyrightable artifacts; the copyleft binding is most secure when these accompany the factual Result rather than the Result being distributed standalone—and empirical work on copyleft propagation in dependency graphs [22, 23] suggests the binding will, in practice, travel with the code.

C. The hardware-attestation gap

The two-tier hierarchy relies on the operational claim that the root key was generated and is stored offline. The toolkit cannot cryptographically verify this: a malicious or compromised issuer can generate a root key on an internet-connected machine, publish a `keys.json` byte-identical to a properly air-gapped one, and the recipient cannot tell the difference from the artifacts alone. Closing the gap technically requires hardware attestation—a TPM/HSM quote of the keygen environment—and is currently blocked on two preconditions: no commodity HSM yet ships ML-DSA or SLH-DSA keygen with an attestation envelope, and no widely-adopted attestation format covers PQ key types. Until those are met, the only mitigation is **structured operational disclosure**: the issuer publishes a written provenance statement alongside the pinned fingerprint, specifying when and how the root key was generated, where its material is stored, who has access, how signing ceremonies are conducted, and whether the procedure has been witnessed or audited. The toolkit recommends this and ships a worked example. Disclosure is not attestation, and we describe it as such; what it provides is a substrate against which an independent reader can assess the plausibility of the process claim and against which the issuer can be held to account if a divergence is later discovered.

D. Threshold custody and adoption friction

SLH-DSA has no native M-of-N construction. Multi-maintainer projects can split the root key with Shamir's Secret Sharing—operationally expensive—or accept single-custody for v1; the spec reserves a v1.1 multi-root (`root_keys: [...]`, any-of) extension. The least technical limitation is the most practical: even with the toolkit, every adopter must operate at least one offline key and produce signed metadata at every release. Our measurements show the *cryptographic* cost is negligible (§VIII), but the *operational* cost—an offline ceremony, key custody, CI integration—is real, and enterprise-migration studies are unanimous that such process change, not algorithm speed, is the binding constraint [8, 9, 58]. The toolkit aims to make compliance trivial, but compliance is not free, and we do not claim otherwise.

E. An operational cost model

It is worth separating the two costs the construction imposes, because they have very different magnitudes. The *per-release* cost is what our evaluation measures: a sub-millisecond signature, a few-millisecond aggregate for a large release, and a bounded metadata premium (§VIII)—negligible, and fully automatable in CI. The *per-issuer* cost is the offline root-key ceremony, and it is where real effort lives. A reasonable cadence is a quarterly `keys.json` re-signing plus an annual key-rotation review; at that rate an issuer performs ∼4–5 offline ceremonies per year, each a sneakernet round trip to an air-gapped machine. The ceremony cost is therefore amortized over an entire quarter of releases rather than paid per release, which is the explicit reason the design pushes the slow, high-assurance SLH-DSA operation to the root and keeps the fast ML-DSA operation on the per-release path. For a solo researcher the ceremony is minutes of process a few times a year; for an institutional issuer it is an audited procedure with split custody, but it is *rare*, *rehearsed*, and off the critical path of any individual release. This asymmetry—cheap and frequent on the signing path, expensive and rare on the root path—is the operational heart of the two-tier model, and it is what makes a mandatory condition tolerable in practice rather than merely in principle.

F. Alternative bindings considered and rejected

The PQ-integrity requirement could in principle be attached to several different carriers, and it is worth saying why the *license* is the right one. *A repository policy*—"Zenodo requires PQ signatures"—binds only that repository: a dataset re-hosted elsewhere, or downloaded and re-shared, sheds the requirement immediately, and no single repository can speak for the whole reuse graph. *A data-use agreement* or click-through contract binds only the parties who sign it, has no transitivity to downstream recipients who never see it, and is precisely the per-recipient-agreement overhead the construction is designed to avoid. *A voluntary convention*—"please sign your releases with PQ keys"—has no propagation mechanism at all and, as the supply-chain literature documents for SBOMs and signing, adoption stalls without an enforceable trigger [17, 18]. *A standards body mandate* moves slowly and binds implementers of the standard, not redistributors of the data. The license is the only carrier that is simultaneously *transitive* (copyleft already propagates to derivatives), *self-enforcing* (breach terminates the grant), and *attached to the artifact rather than to a party or a venue*. That is not an accident of convenience; it is the same reason copyleft was the effective carrier for the source-availability requirement that a mere recommendation never achieved. QGPL reuses a propagation mechanism with a thirty-year track record, and points it at a new obligation.

G. Generalization beyond open science

Although open science is the load-bearing case (§I), the construction is not specific to it. Any domain that combines long retention, reuse across trust boundaries, and an absence of a universally trusted intermediary has the same shape. *Regulated records*—financial, clinical-trial, and legal-evidence archives—are retained for decades and must remain attributable long after the signatures protecting them were generated; PQ digital signatures for medical imaging are already being explored on exactly this long-term-resilience argument [34]. *Machine-learning artifacts*—datasets and model weights distributed openly and re-used as dependencies—face a documented authenticity gap whose existing mitigations are classical and discretionary [43]; a license-carried PQ requirement would propagate integrity obligations through model-derivation chains the way it does through data-derivation chains. *Embedded and blockchain settings*, where long-lived records meet constrained verifiers, are an active site of PQ-signature engineering [37], and the additive-hybrid and offline-root patterns transfer directly. In each case the two clauses—copyleft propagation plus a mandatory quantum-safe condition—are domain-independent; what changes is the artifact, the retention horizon, and the regulatory backdrop, not the mechanism. We scope the present paper to open science because that is where the case is cleanest and the evaluation most directly relevant, but we expect the construction's value to be broadest exactly where classical, infrastructure-dependent integrity is least durable.

H. Co-existence with existing classical signatures

QGPL does not ask a producer to abandon the tooling they already run; it asks them to add a PQ signature that a classical one cannot replace. A release can therefore carry a Sigstore bundle, a detached PGP signature, *and* a QGPL manifest simultaneously, each serving a different constituency: the Sigstore bundle binds an OIDC identity for consumers who trust that log today, the PGP signature serves a legacy web-of-trust, and the QGPL manifest provides the PQ-secure, license-mandated, infrastructure-free guarantee that survives the others' eventual obsolescence. The toolkit's stance toward these is explicit and non-equivalent: it can *read* a Sigstore bundle and *augment* it with a PQ signature so both ship together, but it never treats a classical signature as satisfying §5, because the entire premise is that classical signatures are the ones a CRQC forges. This augment-don't-replace posture matters for adoption: an institution mid-migration need not rip out its classical signing to become QGPL-compliant; it adds the PQ layer alongside and decommissions the classical one on its own schedule, exactly the incremental path the migration literature recommends [8, 9]. The cost of carrying both is additive and small—a second manifest and a second signature—and is dominated, as everywhere in this construction, by the data rather than by the cryptography.

I. Policy, funders, and ecosystem positioning

The construction lands in a policy environment already moving toward mandated data integrity and provenance. Funder and journal requirements increasingly treat FAIR data sharing as a condition of publication, and the governance literature around FAIRness assessment is explicitly about making such obligations checkable rather than aspirational [11, 45]. A mandatory PQ-integrity condition is a natural complement: where a funder mandate says "deposit FAIR data," QGPL adds "… with integrity that survives the deposit's archival horizon," and supplies the mechanism to verify it. The same posture suits government and EU contexts, where long-retention obligations and a preference for sovereignty over third-party trust anchors make an infrastructure-free, PQ-by-default model attractive—and where OSI approval, which QGPL may or may not pursue (§XIII), matters less than demonstrable quantum-safety. We are careful not to overclaim here: QGPL is a technical and legal instrument, not a policy, and its uptake depends on exactly the kind of pilot adoption and institutional buy-in that the migration literature identifies as the rate-limiting step [8, 58]. What the construction offers the policy conversation is a concrete, measured, implementable option where previously there was only a recommendation.

J. When the construction is the wrong tool

Intellectual honesty requires naming the cases where QGPL is a poor fit, because a condition that is mandatory everywhere is credible nowhere. QGPL is the *wrong* tool when: *(i) the artifact is short-lived*—a CI build that is rebuilt nightly gains nothing from a thirty-year integrity horizon, and the fixed metadata premium (§VIII-B) is pure overhead; *(ii) a trusted intermediary already exists and is durable*—inside a single organization with its own PKI and retention guarantees, an internal CA may be simpler than out-of-band root pins; *(iii) the work is purely uncopyrightable fact distributed standalone*, where the copyleft hook is weakest (§XIII) and a contractual or repository-policy mechanism may have more legal traction; *(iv) OSI approval is a hard requirement* for the adopter's ecosystem and the project is unwilling to soften §5 to a recommendation; or *(v) confidentiality, not integrity, is the actual threat*, which is the transport problem PQ-TLS solves and which QGPL explicitly does not address. Recognizing these boundaries is part of the contribution: the construction is sharpest for long-lived, reused, intermediary-free artifacts, and we would rather it be adopted there than mandated indiscriminately. A condition proposed without its own counter-indications invites exactly the backlash that sank earlier over-broad licensing experiments.

XIV. Related Work

The PQ transition and HNDL. A large recent literature characterizes the post-quantum transition as an organizational, multi-year undertaking whose timelines extend

well beyond a decade for large enterprises [8, 9, 28, 58], and formalizes "harvest now, decrypt later" as a temporal risk driven by data longevity [5–7]. This work overwhelmingly addresses *confidentiality*; QGPL targets the underexamined integrity dual, where the relevant longevity is the archival horizon of released data rather than the secrecy lifetime of a message.

PQ signature primitives and hybrids. ML-DSA implementation and acceleration studies show lattice signatures are practical across platforms from ARMv8 to dedicated hardware [35, 36], while hash-based schemes are repeatedly singled out for long-term and archival use on account of their conservative security assumption [33, 34, 37]. A parallel line designs hybrid/composite PQ+classical signatures and keys for the transition window [4, 49–52]; QGPL adopts hybrids but, unlike composite drafts that permit threshold-style semantics, fixes a strictly additive reading and measures its near-zero marginal cost (§VIII-A). Crucially, where these works optimize the *primitive*—smaller signatures, faster verification, better hardware utilization—QGPL is orthogonal: it inherits whatever primitive NIST blesses and asks instead *who is obligated* to use it. A faster ML-DSA or a more compact composite encoding makes QGPL cheaper without changing the construction, which is the point of the algorithm-agility clause; the contribution is the binding, not the cipher. Surveys of PQ-TLS and PQC-in-networks [29, 41, 53, 55, 59] address the *wire*; QGPL is deliberately scoped to artifacts-on-disk and co-exists with PQ transport rather than competing with it.

Quantum-threat estimation and the migration gap. A continuing line of work estimates the quantum resources needed to break RSA and ECC, refining circuit constructions for modular arithmetic and elliptic-curve operations and, in turn, the plausibility window for a CRQC [38, 39]; KEM-resilience and network-feasibility studies complement it on the defensive side [41, 55, 59]. In parallel, a large practitioner-facing literature documents that the binding constraint on post-quantum readiness is organizational rather than algorithmic: cryptographic inventory, dependency analysis, crypto-agility, and multi-year coordination dominate the migration cost [8, 9, 28, 29, 58]. QGPL is positioned squarely in that gap: it does not advance the threat estimate or the primitive, but supplies a concrete, enforceable mechanism that turns "organizations should adopt PQ integrity" into a condition that propagates and self-enforces, which is precisely the step the migration literature finds hardest to achieve voluntarily.

Software-supply-chain integrity. Sigstore [14], in-toto [15], SLSA [16], SBOM practice and tooling [17, 42, 60], and their systematization [18] are the closest infrastructural neighbors; recent work even extends artifact signing to ML datasets and models with explicit attention to verification overhead [43]. We adopt Sigstore's conceptual model of "manifest plus signed key document" while replacing its OIDC-and-transparency-log identity model with an offline-root-plus-published `keys.json`, and—unlike all of the above—we make the signature PQ and *license-mandated* rather than classical and discretionary. SLSA is orthogonal: a release can be both QGPL-compliant and SLSA Level 3, and neither subsumes the other.

TABLE X
POSITIONING AGAINST RECENT ARTIFACT-SIGNING WORK. "PQ" = POST-QUANTUM SIGNING; "DATA" = TARGETS DATASETS/MODELS, NOT ONLY CODE; "LICENSE" = OBLIGATION CARRIED BY A LICENSE; "NO INFRA" = VERIFIABLE WITHOUT CA/LOG/ONLINE SERVICE.

Work	PQ	Data	License	No infra
Sigstore [14]	no	no	no	no
in-toto/SLSA [15, 16]	no	partial	no	partial
Sentry (ML artifacts) [43]	no	yes	no	yes
PQ medical-image sigs [34]	yes	yes	no	yes
Compact hybrid sigs [50]	yes	no	no	yes
QGPL (this work)	**yes**	**yes**	**yes**	**yes**

Research-data integrity and FAIR. The FAIR principles and the infrastructure around them [10–12, 45, 46] make long-lived, reusable data an explicit goal, which is exactly the regime in which release-time classical signatures expire before the data does. QGPL supplies the missing PQ-integrity layer for that regime and uploads cleanly into existing archival flows (Zenodo, Figshare, DOI archives), since manifests are just files.

Licensing as a propagation mechanism. GPL v3 [19], CC share-alike [20], and the ODbL [21] are the closest licensing prior art, and empirical studies of copyleft show its viral condition is precisely what makes obligations propagate—and fail to be met—through reuse graphs [22, 23, 44, 61]. QGPL repurposes that same propagation to carry a cryptographic-integrity obligation, which to our knowledge no prior license does.

XV. Conclusion and Future Work

QGPL v3.0 demonstrates that a copyleft license can be a load-bearing carrier for mandatory post-quantum distribution integrity, without centralized infrastructure and without requiring adopters to trust any party beyond the issuer of the Work. Our evaluation shows the cryptographic cost is small in every dimension that matters to a publisher or recipient: ML-DSA-65 signs and verifies a release in sub-millisecond time; hybrids add tens of bytes and fractions of a millisecond; mandatory metadata falls to parts-per-million of payload for realistic datasets and amortizes to ∼148 bytes per artifact in large releases; the full offline trust chain verifies in ∼2.3 ms; and the verifier detects every defined tampering mode while still permitting explicit PQ-only recovery decades later. The construction is most valuable where archival horizons exceed cryptographic horizons—paradigmatically, open scientific data—and we expect it to be adopted there first.

Concretely, our evaluation establishes:

- **Signing is sub-millisecond.** ML-DSA-65 signs a release in 0.72 ms and verifies in 0.37 ms; the three ML-DSA parameter sets differ by under 1.3×, so higher security is nearly free (§VIII-A, Table V).

- **Hybrids are near-free.** Adding an Ed25519 half costs +64 bytes and <0.3 ms, making defense-in-depth a default rather than a luxury.
- **Mandatory metadata amortizes.** Overhead is $\Theta(1/\text{size})$, below 1% for artifacts $\geq$ 11 KiB and 106 ppm at 100 MiB; an aggregate manifest amortizes to 148 bytes per artifact at N=1000 (§VIII-B, Eq. (1)).
- **Verification is cheap and offline.** The full two-signature trust chain verifies in 2.26 ms with no network, sustaining hundreds of releases per second per core (§VIII-C).
- **Failures are caught; recovery is deliberate.** All six conformance cases produce the spec-mandated verdict, and a hybrid whose classical half is later broken is rejected by default yet recoverable by explicit PQ-only choice.

The recurring theme is asymmetry: the costs that scale with releases and recipients are negligible, while the one genuinely expensive operation—the offline root ceremony—is rare and amortized over a quarter of releases. That is what makes a *mandatory* condition defensible.

The path to v1.0 of the toolkit, and the open problems beyond it, are concrete:

1) **Reference toolkit and slim verifier.** A full CLI over `liboqs-python` and `jcs`, plus a dependency-light `qgpl-verify` carrying a pure-software ML-DSA verifier so reviewers and archival verifiers can check releases without compiling a PQ stack.
2) **Self-signed bootstrap.** The toolkit's own first production release will be PQ-signed with the generator of §X, demonstrating the construction by use.
3) **Archive integration.** Turn-key patterns for Zenodo, Figshare, and DOI-archive workflows, and a GitHub Action that signs a release glob and uploads `*.qgpl.json`/`*.qgpl.sig` as assets.
4) **Pilot releases.** One or two QGPL-licensed dataset releases from partner institutions, to exercise the construction against real archival pipelines before generalizing.
5) **Hardware root attestation.** Close the ğXIII gap by carrying a TPM/HSM keygen quote in a `public_key.attestation` field; blocked on commodity HSMs shipping ML-DSA/SLH-DSA keygen with an attestation envelope and on a standard attestation format covering PQ key types.
6) **Threshold custody.** A multi-root (`root_keys`, any-of) or documented Shamir-split workflow, removing the single-custody restriction that currently limits multi-maintainer institutional adoption.
7) **Transparency option.** A PQ-signed, Rekor-equivalent inclusion proof in the reserved `timestamp.transparency_log` slot, for issuers who do want public auditability without sacrificing the offline default.

The most important of these are hardware root attestation and threshold custody (both technical) and the OSI/FSF approval discussion (a governance choice); we invite scrutiny and counterargument on all three.

Availability

QGPL v3.0 license text, the Compliance Toolkit Specification (Draft v0.5.1), the conformance fixture, the reference implementation and experiment harness (`docs/paper/experiments/`), and the source of this paper are maintained at https://github.com/Grigaliunas/qgpl under QGPL v3.0. Every figure and numeric result is regenerated from the committed `results.json`. A frozen citable snapshot will be deposited at Zenodo on the first stable release of the v1.0 toolkit.

Acknowledgment

The author thanks The Open Reality Initiative for hosting the QGPL project and the reviewers of the toolkit specification drafts for sharpening the hardware-attestation and threshold-custody discussions.

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1994, pp. 124–134.

[2] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?" *IEEE Security & Privacy*, vol. 16, no. 5, pp. 38–41, 2018.

[3] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.

[4] F. Driscoll, M. Parsons, and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes," RFC 9794, Jun. 2025. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9794.html

[5] F. Kagai, P. Branch, and J. But, "Harvest-now, decrypt-later: A temporal cybersecurity risk in the quantum transition," *Telecom*, vol. 6, no. 4, p. 100, 2025. [Online]. Available: https://doi.org/10.3390/telecom6040100

[6] J. Mascelli and M. Rodden, ""harvest now decrypt later": Examining post-quantum cryptography and the data privacy risks for distributed ledger networks," Board of Governors of the Federal Reserve System, Finance and Economics Discussion Series 2025-093, 2025. [Online]. Available: https://doi.org/10.17016/FEDS.2025.093

[7] J. Blanco-Romero, F. Almenares Mendoza, and C. García Rubio, "On the practical feasibility of harvest-now, decrypt-later attacks," arXiv preprint arXiv:2603.01091, 2026. [Online]. Available: https://doi.org/10.48550/arXiv.2603.01091

[8] R. Campbell, "Enterprise migration to post-quantum cryptography: Timeline analysis and strategic frameworks," *Computers*, vol. 15, no. 1, p. 9,

2025. [Online]. Available: https://doi.org/10.3390/computers15010009

[9] K. F. Hasan, L. Simpson, M. A. R. Baee *et al.*, "A framework for migrating to post-quantum cryptography: Security dependency analysis and case studies," *IEEE Access*, vol. 12, pp. 23 427–23 450, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3360412

[10] M. D. Wilkinson, M. Dumontier, I. J. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne *et al.*, "The FAIR guiding principles for scientific data management and stewardship," *Scientific Data*, vol. 3, p. 160018, 2016.

[11] M. D. Wilkinson, S.-A. Sansone, E. Méndez *et al.*, "Community-driven governance of FAIRness assessment: an open issue, an open discussion," *Open Research Europe*, vol. 2, p. 146, 2022. [Online]. Available: https://doi.org/10.12688/openreseurope.15364.1

[12] FORCE11, "Joint Declaration of Data Citation Principles," 2014. [Online]. Available: https://doi.org/10.25490/a97f-egyk

[13] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. Association for Computing Machinery, 1996, pp. 212–219.

[14] Z. Newman, J. S. Meyers, S. Torres-Arias *et al.*, "Sigstore: Software signing for everybody," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. Association for Computing Machinery, Nov. 2022, pp. 2353–2367. [Online]. Available: https://doi.org/10.1145/3548606.3560596

[15] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Cappos, "in-toto: Providing farm-to-table guarantees for bits and bytes," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1393–1410. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias

[16] OpenSSF, "Supply-chain Levels for Software Artifacts (SLSA)," Project specification and guidance, 2026, accessed 2026-05-24. [Online]. Available: https://slsa.dev/

[17] B. Xia, T. Bi, Z. Xing *et al.*, "An empirical study on software bill of materials: Where we stand and the road ahead," arXiv preprint arXiv:2301.05362, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2301.05362

[18] C. Okafor, T. R. Schorlemmer, and S. Torres-Arias, "SoK: Analysis of software supply chain security by establishing secure design properties," pp. 15–24, Nov. 2022. [Online]. Available: https://doi.org/10.1145/3560835.3564556

[19] Free Software Foundation, "GNU General Public License, Version 3," Jun. 2007. [Online]. Available: https://www.gnu.org/licenses/gpl-3.0.html

[20] Creative Commons, "Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)," 2013. [Online]. Available: https://creativecommons.org/licenses/by-sa/4.0/

[21] Open Knowledge Foundation, "Open Database License (ODbL) v1.0," 2009. [Online]. Available: https://opendatacommons.org/licenses/odbl/1-0/

[22] R.-H. Pfeiffer, "License incompatibilities in software ecosystems," arXiv preprint arXiv:2203.01634, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2203.01634

[23] S. Qiu, D. M. German, and K. Inoue, "Empirical study on dependency-related license violation in the JavaScript package ecosystem," *Journal of Information Processing*, vol. 29, pp. 296–304, 2021. [Online]. Available: https://doi.org/10.2197/ipsjjip.29.296

[24] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," FIPS PUB 203, Aug. 2024. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.203

[25] ——, "Module-Lattice-Based Digital Signature Standard," FIPS PUB 204, Aug. 2024. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.204

[26] ——, "Stateless Hash-Based Digital Signature Standard," FIPS PUB 205, Aug. 2024. [Online]. Available: https://doi.org/10.6028/NIST.FIPS.205

[27] D. Cooper, D. Apon, Q. Dang, M. Davidson, M. Dworkin, and C. Miller, "Recommendation for Stateful Hash-Based Signature Schemes," National Institute of Standards and Technology, NIST Special Publication 800-208, Oct. 2020. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-208

[28] K. Cherkaoui Dekkaki, I. Tasic, and M.-D. Cano, "Exploring post-quantum cryptography: Review and directions for the transition process," *Technologies*, vol. 12, no. 12, p. 241, 2024. [Online]. Available: https://doi.org/10.3390/technologies12120241

[29] E. Zeydan, Y. Türk, B. Aksoy *et al.*, "Recent advances in post-quantum cryptography for networks: A survey," in *2022 Seventh International Conference on Mobile and Secure Services (MobiSecServ)*. IEEE, 2022, pp. 1–8. [Online]. Available: https://doi.org/10.1109/MobiSecServ50855.2022.9727214

[30] J. W. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber: A cca-secure module-lattice-based kem," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 353–367.

[31] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium: A lattice-based digital signature scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 1, pp. 238–268, 2018.

[32] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: Practical stateless hash-based signatures," in *Advances in Cryptology – EUROCRYPT 2015*. Springer, 2015, pp. 368–397.

[33] L. Li, X. Lu, and K. Wang, "Hash-based signature revisited," *Cybersecurity*, vol. 5, no. 1, 2022. [Online]. Available: https://doi.org/10.1186/s42400-022-00117-w

[34] K. S. Roy, S. Singh, and P. P. Srivathsa, "Post-quantum digital signatures for enhanced medical image security," *IET Quantum Communication*, vol. 6, no. 1, 2025. [Online]. Available: https://doi.org/10.1049/qtc2.70006

[35] Y. Kim, J. Song, and T.-Y. Youn, "Crystals-dilithium on ARMv8," *Security and Communication Networks*, vol. 2022, p. 5226390, 2022. [Online]. Available: https://doi.org/10.1155/2022/5226390

[36] Q. Truong, P. Duong-Ngoc, and H. Lee, "Efficient low-latency hardware architecture for module-lattice-based digital signature standard," *IEEE Access*, vol. 12, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3370470

[37] D. Marchsreiter, "Towards quantum-safe blockchain: Exploration of PQC and public-key recovery on embedded systems," *IET Blockchain*, vol. 5, no. 1, 2025. [Online]. Available: https://doi.org/10.1049/blc2.12094

[38] X. Liu, H. Yang, and L. Yang, "Feasibility analysis of cracking RSA with improved quantum circuits of the shor's algorithm," *Security and Communication Networks*, vol. 2023, p. 2963110, 2023. [Online]. Available: https://doi.org/10.1155/2023/2963110

[39] H. T. Larasati, D. S. C. Putranto, R. W. Wardhani *et al.*, "Depth optimization of FLT-based quantum inversion circuit," *IEEE Access*, vol. 11, pp. 54 910–54 927, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3280632

[40] M. Mosca and M. Piani, "Quantum Threat Timeline Report 2023," Global Risk Institute, Tech. Rep., Dec. 2023. [Online]. Available: https://globalriskinstitute.org/publication/2023-quantum-threat-timeline-report/

[41] S. Farooq, A. Altaf, F. Iqbal *et al.*, "Resilience optimization of post-quantum cryptography key encapsulation algorithms," *Sensors*, vol. 23, no. 12, p. 5379, 2023. [Online]. Available: https://doi.org/10.3390/s23125379

[42] Ö. Kağızmandere and H. Arslan, "Vulnerability analysis based on SBOMs: A model proposal for automated vulnerability scanning for CI/CD pipelines," *International Journal of Information Security Science*, vol. 13, no. 2, pp. 33–42, 2024. [Online]. Available: https://doi.org/10.55859/ijiss.1455039

[43] A. Gan and Z. Ghodsi, "Sentry: Authenticating machine learning artifacts on the fly," in *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25)*. ACM, 2025, pp. 3550–3563. [Online]. Available: https://doi.org/10.1145/3719027.3765070

[44] N. Harutyunyan, "Open source software governance: Distilling and applying industry best practices," in *Open Source Systems*. Springer, 2022, pp. 73–90. [Online]. Available: https://doi.org/10.1007/978-3-030-83128-8_5

[45] C. R. Kirkpatrick, K. Coakley, J. Christopher *et al.*, "Engaging with researchers and raising awareness of FAIR and open science through the FAIR+ implementation survey tool (FAIRIST)," *Data Science Journal*, vol. 22, p. 32, 2023. [Online]. Available: https://doi.org/10.5334/dsj-2023-032

[46] Z. Goldblum, Z. Xu, H. Shi *et al.*, "Pennsieve: A collaborative platform for translational neuroscience and beyond," *Scientific Data*, vol. 12, no. 1, 2025. [Online]. Available: https://doi.org/10.1038/s41597-025-06075-5

[47] The Open Reality Initiative, "QGPL Compliance Toolkit Specification (draft v0.5.1)," Repository specification, 2026, docs/COMPLIANCE-TOOLKIT-SPEC.md. [Online]. Available: https://github.com/Grigaliunas/qgpl

[48] A. Rundgren, B. Jordan, and S. Erdtman, "JSON Canonicalization Scheme (JCS)," RFC 8785, Jun. 2020. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8785.html

[49] S. Ricci, P. Dobiáš, L. Malina *et al.*, "Hybrid keys in practice: Combining classical, quantum and post-quantum cryptography," *IEEE Access*, vol. 12, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3364520

[50] H.-Y. Kwon, I. Bajuna, and M.-K. Lee, "Compact hybrid signature for secure transition to post-quantum era," *IEEE Access*, vol. 12, 2024. [Online]. Available: https://doi.org/10.1109/ACCESS.2024.3374645

[51] J. Müller and J. Oupický, "Post-quantum XML and SAML single sign-on," *Proceedings on Privacy Enhancing Technologies*, no. 2024(4), 2024. [Online]. Available: https://doi.org/10.56553/popets-2024-0128

[52] M. Ounsworth, J. Gray, M. Pala, J. Klaußner, and S. Fluhrer, "Composite Module-Lattice-Based Digital Signature Algorithm (ML-DSA) for use in X.509 Public Key Infrastructure," Internet-Draft draft-ietf-lamps-pq-composite-sigs-19, Apr. 2026, work in progress, revision 19. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-lamps-pq-composite-sigs/

[53] N. Alnahawi, J. Müller, J. Oupický *et al.*, "A comprehensive survey on post-quantum TLS," *IACR Communications in Cryptology*, vol. 1, no. 2, 2024. [Online]. Available: https://doi.org/10.62056/ahee0iuc

[54] R. Aroshas, "QuantCrypt: Cross-platform Python library for Post-Quantum cryptography using precompiled PQClean binaries," 2025, v1.0.1; wraps NIST

FIPS 203/204/205 reference parameter sets. [Online]. Available: https://pypi.org/project/quantcrypt/

[55] I. Tzinos, K. Limniotis, and N. Kolokotronis, "Evaluating the performance of post-quantum secure algorithms in the TLS protocol," *Journal of Surveillance, Security and Safety*, vol. 3, 2022. [Online]. Available: https://doi.org/10.20517/jsss.2022.15

[56] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila, "Transitioning to a quantum-resistant public key infrastructure," in *Post-Quantum Cryptography*. Springer, 2017, pp. 384–405.

[57] The Open Reality Initiative, "QGPL Compliance Toolkit Conformance Fixture (spec-0.4)," Repository conformance fixture, 2026, docs/conformance/spec-0.4/. [Online]. Available: https://github.com/Grigaliunas/qgpl

[58] M. S. D. Thakur, K. Vidhani, and H. B. Syed, "Enterprise post quantum cryptography migration tools," in *2024 16th International Conference on Communication Systems and Networks (COMSNETS)*. IEEE, 2024, pp. 327–329. [Online]. Available: https://doi.org/10.1109/COMSNETS59351.2024.10427442

[59] B. Lonc, A. Aubry, H. Bakhti *et al.*, "Feasibility and benchmarking of post-quantum cryptography in the cooperative ITS ecosystem," in *2023 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2023, pp. 215–222. [Online]. Available: https://doi.org/10.1109/VNC57357.2023.10136335

[60] X. Zhou, J. Xu, X. Li *et al.*, "A blockchain-based and microservices-architected software composition analysis system," *Journal of Software: Evolution and Process*, vol. 36, no. 10, 2024. [Online]. Available: https://doi.org/10.1002/smr.2675

[61] M. Papoutsoglou, G. M. Kapitsaki, and D. M. German, "An analysis of open source software licensing questions in Stack Exchange sites," arXiv preprint arXiv:2110.00361, 2021. [Online]. Available: https://doi.org/10.48550/arXiv.2110.00361