

From Signature Generation to Operational Verification: Deployment Lessons from Hybrid Post-Quantum Software Signing in CI/CD

Rafael Lucas Silva Luiz Otavio Duarte Larissa de Oliveira Figueira
Facti – Fundação de Apoio à Capacitação em Tecnologia da Informação, Brazil
`{rafael.silva,larissa.oliveira,luiz.duarte}@facti.com.br`

1. Deployment Experience

Software supply chains rely extensively on digital signatures to establish the authenticity and integrity of packages, releases, container images, manifests, and other artifacts. Unlike ephemeral authentication exchanges, software signatures may need to remain verifiable offline or years after an artifact was originally produced.

We investigated post-quantum software signing through a practitioner-oriented DevSecOps implementation in GitLab CI/CD. The pipeline combines Shift-Left security controls, deterministic artifact manifest generation, and hybrid signature generation.

At the signing stage, a deterministic manifest containing SHA-256 hashes of the relevant artifact files is generated. Its digest is independently signed through two paths: a classical signature managed through HashiCorp Vault Transit and a post-quantum signature from the Dilithium family implemented using Open Quantum Safe tooling. The signatures, algorithm identifiers, manifest digest, and associated metadata are then aggregated into a structured signature bundle. This design is hybrid at the verification-policy layer rather than a composite signature construction. Both signatures remain independently generated and verifiable. Existing consumers may therefore continue using the classical path, while PQ-capable consumers can additionally validate the post-quantum signature.

The prototype used Dilithium-family implementations through `liboqs` and should be interpreted as an experimental migration implementation rather than a complete production deployment of the standardized ML-DSA ecosystem. Nevertheless, the observed issues around verification, metadata, key management, and migration policy remain directly relevant to ML-DSA adoption.

Adding the PQ signature operation itself was technically straightforward. In one evaluated configuration, the pipeline executed on a GitLab Runner with 4 vCPUs, 8 GB RAM, Ubuntu 24.04, and an artifact of approximately 12 MB. Table 1 summarizes the measured overhead.

Table 1: Representative signing overhead.

Operation	Time
Manifest generation	0.08 s
Classical signature	0.12 s
Dilithium signature	0.34 s
Signature bundle creation	0.02 s
Total additional time	0.56 s

For CI/CD workflows whose execution time is typically measured in minutes, this latency is small. In our scenario, signature computation was therefore not the dominant obstacle to initial PQ adoption. The harder problems appeared after signature generation, when authentication material had to be interpreted, governed, distributed, and verified.

2. What Was Hard: From Signatures to Authentication

Representation and verification semantics. Producing a bundle containing two signatures is straightforward, but making it interoperable across independent consumers is harder. A verifier needs to know not only how to validate each signature, but also what successful validation means. Must both signatures validate? Can either establish trust? What happens when the PQ algorithm is unsupported? Which metadata is bound to the signatures? How are algorithm transition and deprecation represented? A proprietary bundle can encode these rules locally, but this moves complexity from cryptographic implementation into verification policy. Producing two signatures is therefore different from providing an interoperable hybrid authentication mechanism.

Key management. The classical path relied on an established Vault-managed key lifecycle, whereas the PQ component required a separate integration path. This creates operational asymmetry. Production PQ signing requires equivalent guarantees for key generation, custody, access control, rotation, revocation, backup, recovery, and auditing. Integration with HSMs, PKCS#11 interfaces, centralized key-management platforms, and organizational trust infrastructures may influence adoption more strongly than raw signature performance.

Consumer-side migration. CI/CD is a convenient point for introducing PQ signatures because producers control their own build pipelines. Consumers are heterogeneous. Registries, deployment systems, artifact validators, package managers, transparency systems, offline tools, and third parties may adopt PQ verification at different speeds. Our

implementation therefore preserves the classical signature as the compatibility path while carrying the PQ signature as additional verification evidence. This enables incremental adoption, but leaves an important policy question: when should PQ verification move from optional evidence to a mandatory trust condition?

Long-lived verification. Software and firmware artifacts may need to remain verifiable after signing keys have rotated, certificates have expired, algorithms have been deprecated, or the original signing infrastructure has disappeared. Long-lived authentication therefore requires more than preservation of the signature value. Verification evidence may also need algorithm identifiers, public-key or certificate information, timestamps, provenance metadata, revocation evidence, and the verification policy applicable when the artifact was produced. Our current bundle supports artifact integrity and dual signatures, but does not completely solve this archival-validation problem.

3. Position and Open Questions

Our experience suggests that the next bottleneck for post-quantum software authentication is not primarily the ability to compute ML-DSA signatures. The harder challenge is defining interoperable operational practices around them.

We identify four deployment questions:

1. **What should a deployable hybrid software-signing profile look like?** Implementers need common semantics for binding classical and PQ signatures to the same artifact and expressing verification requirements.
2. **How should verification policy evolve during migration?** Systems need explicit transitions from classical-only to classical-plus-PQ and eventually PQ-authoritative verification, without ambiguous acceptance rules.
3. **What evidence must accompany signatures for offline and archival validation?** Long-lived artifacts need verification models that survive key rotation, certificate expiration, revocation, algorithm migration, and infrastructure changes.
4. **How should existing software supply-chain mechanisms integrate PQ authentication?** Practical deployment must coexist with SBOMs, provenance attestations, transparency mechanisms, secure update frameworks, package signing, and container-signing workflows.

These questions indicate that deployment guidance should address not only algorithm identifiers and cryptographic encodings, but also verification semantics, migration states, and the lifecycle of authentication evidence.

4. Conclusion

Our implementation indicates that post-quantum software signing can be introduced incrementally into existing CI/CD systems with low computational cost and without immediately replacing classical verification. However, generating a PQ signature should not be confused with deploying post-quantum authentication. The harder transition occurs at the boundaries between signing systems, key-management infrastructure, artifact formats, verification policy, consumers, and long-term validation mechanisms. We therefore argue that deployment work should increasingly move from algorithm availability toward deployable authentication profiles and migration semantics. Hybrid deployment provides time for the ecosystem to evolve; the remaining challenge is defining how this coexistence becomes interoperable, auditable, and eventually removable once PQ authentication becomes the normal verification path.

References

- [1] National Institute of Standards and Technology, *FIPS 204: Module-Lattice-Based Digital Signature Standard*, 2024.
- [2] R. L. Silva, “Toward Quantum-Resilient Software Supply Chains: A DevSecOps Case Study with Hybrid Post-Quantum Artifact Signing,” in *2026 International Conference on Quantum Communications, Networking, and Computing (QCNC)*, Kobe, Japan, 2026, p. 371. doi: 10.1109/QCNC69040.2026.00064.
- [3] Open Quantum Safe Project, *liboqs: C Library for Quantum-Resistant Cryptographic Algorithms*. Available: <https://openquantumsafe.org/>
- [4] M. Ounsworth et al., “Composite Module-Lattice-Based Digital Signature Algorithm (ML-DSA) for use in X.509 Public Key Infrastructure,” IETF Internet-Draft, [draft-ietf-lamps-pq-composite-sigs](#).