

Beyond Algorithm Choice: Initial Lessons from Preparing a Large Cloud Proxy Service for Post-Quantum Authentication

Harpreet Singh, Zscaler

Yaroslav Rosomakho, Zscaler

Post-quantum key establishment is already moving into deployment, but post-quantum authentication raises a different set of operational challenges. This paper summarises several early lessons from preparing a large, globally distributed cloud proxy service for migration to post-quantum authentication.

Our work has included evaluation of ML-DSA, SLH-DSA and composite signatures across multiple cryptographic libraries and processor architectures, together with practical questions around key representation and application integration. The measurements cover 34 signature algorithms, six cryptographic implementations and eight hardware platforms.

Performance depends heavily on implementation and hardware

One of the clearest observations is that the performance of a post-quantum signature scheme cannot be characterised independently of the implementation and hardware.

For ML-DSA, newer OpenSSL versions with architecture-specific optimisations provide approximately 3.5–4x higher throughput on several recent Intel Xeon processors than OpenSSL 3.5, while the improvement is much smaller on an older Xeon E5-2637 v2. AWS-LC shows a similar pattern. Across the dataset, the difference between the fastest and slowest ML-DSA implementation on the same processor is approximately fourfold in the median case.

Hardware classification is also a poor predictor of performance. A Raspberry Pi 5 can outperform decade-old Xeon server hardware on several hash-intensive post-quantum workloads, particularly SLH-DSA. Apple M-series processors perform especially well on ML-DSA and SLH-DSA when implementations take advantage of the architecture's SHA-3/SHAKE capabilities.

The practical lesson is that benchmark results obtained from generic or early implementations can substantially overstate the eventual CPU cost of post-quantum deployment.

ML-DSA CPU cost may not be the main problem

ML-DSA is not necessarily computationally expensive compared with classical alternatives. In our measurements, ML-DSA signing can be several times faster than RSA signing, while ML-DSA key generation is orders of magnitude cheaper than RSA key generation.

The more important deployment issue may instead be size. ML-DSA signatures range from approximately 2.4 kB to 4.6 kB, with similarly large public keys. For a cloud proxy service, this affects certificate chains, identity tokens, protocol limits, storage and network fragmentation.

This suggests that migration planning should not focus only on cryptographic throughput. Protocol and implementation assumptions built around small classical signatures may become the more significant constraint.

ML-DSA seed should be the interoperability boundary

A less visible deployment issue is the representation of ML-DSA private keys.

Implementations expose ML-DSA private material differently. Some support expanded private keys, while others, including BoringSSL- and RustCrypto-based implementations we evaluated, operate from the 32-byte ML-DSA seed and do not provide an interchangeable expanded-private-key representation.

Persisting an implementation-specific expanded key therefore creates unnecessary coupling between key storage and a particular cryptographic library.

Our recommendation is:

Where ML-DSA private key material is stored or exchanged independently of a specific implementation, the standard ML-DSA seed should be used as the canonical representation rather than an expanded private key or combined seed-plus-private-key representation.

The expanded key can be generated locally where required. This improves portability between implementations, reduces storage requirements and provides a stable interoperability boundary.

Composite signatures have predictable but asymmetric cost

The measured computational cost of ML-DSA composite signatures is approximately the sum of the costs of their component algorithms, with little additional composite-specific overhead.

The impact is asymmetric, however. Adding a classical algorithm often adds relatively little to signing cost because ML-DSA dominates, while verification can become substantially more expensive. This matters for services that verify signatures at much greater scale than they generate them.

Key lifecycle also matters. For ML-DSA, ECDSA and EdDSA, repeatedly constructing or importing key objects can materially reduce throughput. Migration benchmarks should therefore distinguish between reusable and single-use keys and, where possible, measure complete application operations rather than only primitive cryptographic calls.

Initial conclusions

Our early experience suggests that post-quantum authentication migration is not simply an algorithm substitution exercise.

Three issues appear particularly important:

- **Implementation maturity and hardware acceleration strongly influence CPU cost.** The same algorithm can differ in performance by several times depending on the selected library and processor capabilities.
- **For ML-DSA, object size may be a more significant operational challenge than cryptographic computation.**
- **Private-key representation needs to be treated as an interoperability concern.** Using the ML-DSA seed as the canonical stored and exchanged representation avoids unnecessary dependency on implementation-specific expanded formats.

The measurements supporting these observations are available at <https://signatures.yaroslav.uk/>.

These initial results suggest that successful migration to post-quantum authentication will require attention not only to algorithm selection, but also to cryptographic APIs, key formats, processor-specific optimisation and protocol assumptions. Addressing these deployment interfaces early may remove significant obstacles to broader adoption.