

Beyond Algorithm Choice



Beyond Algorithm Choice

The algorithm is the least interesting variable

Harpreet Singh · Yaroslav Rosomakho — Zscaler

Preparing a global cloud proxy service for post-quantum authentication

11,360 measurements · 8 platforms · 6 libraries · 34 algorithms · <https://signatures.yaroslav.uk/>

Every inspected transaction costs us one or two signatures

Multi-tenant **cloud proxy**, globally distributed — **TLS inspection at scale**



So: how fast is ML-DSA signing?

4–5x

between the fastest and the slowest ML-DSA
implementation — on the **same CPU**

Median over eight CPUs, for each of ML-DSA-44, -65 and -87, signing, single core.

Same algorithm, same silicon. Only the library changed.

Significant signing performance divergence

ML-DSA-65 sign, ops/s, single core — shown because it is the middle (category 3) parameter set;

ML-DSA-44 and -87 give the same picture (27.6x and 27.9x end to end).

	OpenSSL 3.5	OpenSSL 4.1	AWS-LC	BoringSSL	wolfSSL	RustCrypto	spread
Raspberry Pi 5	548	1,226	3,273	1,794	2,510	1,194	6.0x
Apple M4 Max	2,604	4,169	15,954	6,368	6,864	7,625	6.1x
Xeon E5-2637 v2	771	1,167	1,809	1,658	1,961	1,506	2.5x
Xeon 6246	929	2,981	4,525	1,748	2,212	1,689	4.9x
Xeon 6442Y	1,101	4,063	4,528	1,838	2,332	1,787	4.1x
Xeon 6738P	1,555	6,247	7,368	3,560	3,994	4,179	4.7x
Xeon 6761P	1,560	5,757	6,557	2,616	3,250	2,469	4.2x
Xeon 6952P	1,359	5,162	5,830	2,303	2,868	2,280	4.3x

29x fastest to slowest cell · **2.5–6.1x** within one CPU · boxed = per-CPU winner

Where the divergence comes from

OpenSSL 4.1 over OpenSSL 3.5, ML-DSA-65 sign — the gain tracks the *instruction set*, not the version:

AVX2 / AVX-512 Xeons	Xeon E5-2637 v2 (pre-AVX2)	Apple M4 Max	Raspberry Pi 5
3.2 – 4.1x	1.5x	1.6x	2.2x

- **AWS-LC** ships **mldsa-native** — formally verified, CBMC for C plus HOL Light for AArch64 and x86-64 assembly
- **OpenSSL 4.1** ships **its own** AVX2 NTT and AVX-512 SHAKE-x4 assembly → **x86-64 only**
- On M4 Max, OpenSSL 4.1 reaches **0.67x** of a Xeon 6738P. The Mac is not slow — the code path is missing

Verified, fastest, and portable are three different libraries today. Pick two.

And hardware class does not predict it either

Best library on each platform, sign ops/s — a £60 board against a 2013 server:

	Raspberry Pi 5	Xeon E5-2637 v2	
RSA-2048	514	989	Xeon 1.9x
ECDSA P-256	21,161	33,716	Xeon 1.6x
ML-DSA-65	3,273	1,961	Pi 1.7x
SLH-DSA-SHA2-128f	130	42	Pi 3.1x

- The Pi loses **every** classical benchmark, then **wins 17 of 34** — every win is PQ or hash-based
- x86 has **SHA-NI for SHA-2 and no SHA-3 instruction**; Apple M-series has the **ARMv8.2 SHA-3 extension**
- Consequence: SLH-DSA-**SHAKE** runs at **0.19x** of the **SHA-2** parameter sets on modern x86

Your classical capacity model will mis-predict your post-quantum capacity.

Lessons learned from working on the benchmarks

ML-DSA private keys — one key, three legal encodings: `seed` (32 B) · `expandedKey` · `both`

- **BoringSSL and RustCrypto** accept the **seed only**; an expanded key welds the key store to **one library**
- **Recommendation**: store and exchange the **32-byte seed** and expand locally — everyone can read it

SLH-DSA — thin coverage, and the subsets do not line up

- **AWS-LC: none**. BoringSSL: **2 of 12** — SHA2-128s and SHAKE-256f
- JOSE/COSE standardises a *different* two — SHA2-128s and **SHAKE-128s** — so only one is in both

Composites — many of them, and three working groups pick three different sets

- LAMPS **18** · TLS **16** · JOSE/COSE **6**. We measured 6; BoringSSL and AWS-LC skip ML-DSA-87 + Ed448

draft-ietf-cose-sphincs-plus · draft-ietf-lamps-pq-composite-sigs · draft-reddy-tls-composite-mldsa · draft-ietf-jose-pq-composite-sigs

A migration can pass every benchmark and still not be portable.

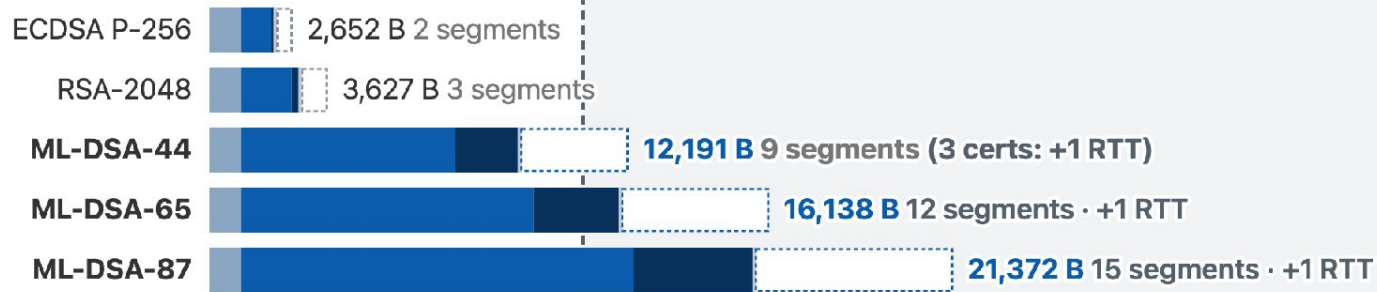
Then we costed it — and the cycles were not the problem

The bytes are one TLS 1.3 handshake — everything the server sends before the client can reply.

Certificates are real DER, issued with OpenSSL 3.6; X.509 adds ~0.5 kB to each one whatever the algorithm.

■ Hello + ML-KEM share ■ leaf + intermediate ■ CertificateVerify ■ EE, Finished, framing □ + a 3rd certificate

initial congestion window: $10 \times 1,460 \text{ B} = 14,600 \text{ B}$



X25519MLKEM768 key share · DER certificates issued by OpenSSL 3.6 · 1,460 B per TCP segment · dashed = with a third certificate

Where 16,138 B comes from (ML-DSA-65): ServerHello + ML-KEM share **1,221** · two DER certificates **11,456** (5,758 leaf + 5,680 intermediate + framing) · CertificateVerify **3,317** · EncryptedExtensions, Finished, records **144**. At 1,460 B per TCP segment → **12 segments**.

The dashed line is the initial congestion window: the server sends ~10 segments, then waits for the client's ACK. ECDSA needs 2, ML-DSA-65 needs 12 — the last 2 arrive **one round trip later**.

Three asks

1 · Make the 32-byte seed the interoperability boundary for ML-DSA private keys.

Store and exchange the seed, expand locally. It is the one encoding everybody can already read.

2 · The community would benefit from a maintained, vendor-neutral signature benchmark.

Ours is a side project on a personal domain. A library × platform × algorithm decision this expensive deserves something citable. We will contribute hardware, results and time.

3 · Treat bytes as the headline cost.

Merkle Tree Certificates and chain compression will buy more than any implementation we measured.

And if you maintain one of these: SLH-DSA in AWS-LC · a verified AArch64 ML-DSA path in OpenSSL · SHA-3 instructions in server silicon.

Everything that cost us time was an interface — a library, a build flag, a key encoding, a protocol limit. Not an algorithm.

Thanks

Zscaler and the other trademarks listed at <https://www.zscaler.com/legal/trademarks> are either (i) registered trademarks or service marks or (ii) trademarks or service marks of Zscaler, Inc. in the United States and/ or other countries. Any other trademarks are the properties of their respective owners.

©2026 Zscaler, Inc. All rights reserved.

