§ FIPS 203 AND 204 · THE BYTE MODEL, MTC, AUTHKEM

What is specific to post-quantum

Supporting document for *"TLS 1.3 never tells the server what a resumed connection must re-check"*, IAB Workshop on Accelerating the Deployment of Post-Quantum Authentication. León Acosta, QuantaKrypto. v1, 28 August 2026.
The byte model with per-row derivations, what post-quantum multiplies and what it does not, Merkle Tree Certificates, and AuthKEM.

1. Provenance for the post-quantum section

The paper's cost table, section 2, rests on three claims. Each traces here.

1.1 The byte figures

Post-quantum, constructed. The script is `construct-chain.py` and its parameters are recorded, so the chain is stated as built rather than inferred from its own totals:

COMPONENT	BYTES	COMPOSITION
leaf certificate	4,635	14 B subject + 1,312 B **ML-DSA-44** public key + 3,309 B ML-DSA-65 signature from the intermediate
intermediate	5,276	15 B subject + 1,952 B **ML-DSA-65** public key + 3,309 B ML-DSA-65 signature from the root
two SCTs	4,840	2 × 2,420 B ML-DSA-44 signatures
CertificateVerify	2,420	ML-DSA-44 signature under the leaf key
total	17,171	leaf ML-DSA-44, intermediate and root ML-DSA-65

This is a construction, not an observation, and it is weaker than an observation in two specific ways. No such chain was retrieved from a live server, and the two ML-DSA-signed SCTs assume a Certificate Transparency design that is not settled. More importantly, **the components are key-plus-signature-plus-name sums and encode no X.509 structure**: the subject fields are 14 and 15 bytes, where real DER certificates carry serial numbers, two AlgorithmIdentifiers, full distinguished names, validity windows and extension sets, typically several hundred bytes each. The classical comparison figures are real DER bytes from `openssl s_client`. So the two sides are not measured the same way, and the asymmetry runs *against* this paper's own argument: adding realistic encoding to the post-quantum side would widen a ratio that already exceeds fivefold, not narrow it.

A correction, and a caution about deriving parameters from totals. Review suggested the intermediate must be ML-DSA-44-keyed, reasoning that an ML-DSA-65 key would sign the leaf with 3,309 bytes and leave only 14 bytes of the leaf for everything else. The subtraction is right and its premise is wrong: there are no certificate-overhead bytes here to reason about, and 14 is not a residue but the literal length of `CN=example.com`. The script says otherwise and the script is authoritative. The lesson is the one the review itself drew: for a chain you constructed, read the construction.

Classical, measured. Retrieved 25 August 2026 with `openssl s_client -showcerts`, DER bytes: ietf.org 3,393 B (4 certificates), github.com 2,721 B (3), cloudflare.com 2,493 B (3), all ECDSA P-256, plus a P-256 CertificateVerify of about 72 B.

1.1b The byte model, derived per row

One constant set, a visible sum per cell, and the same framing on every row except for the key share it carries: 1,088 B of ML-KEM-768 ciphertext post-quantum against 32 B of X25519 classically. Every row counts its CertificateVerify as certificate material.

CONSTANT	VALUE	PROVENANCE

ML-DSA-44 public key / signature	1,312 / 2,420	specified, FIPS 204 Table 2
ML-DSA-65 public key / signature	1,952 / 3,309	specified, FIPS 204 Table 2
ML-KEM-768 ciphertext	1,088	specified, FIPS 203 Table 3
X.509 encoding, leaf / intermediate	660 / 500	**measured**, live letsencrypt.org and cloudflare.com chains
SCT structure, excluding the log signature	53	computed, RFC 6962 section 3.2
X25519 key share, classical rows	32	specified, RFC 8446 section 4.2.8.2. It replaces the 1,088 B ML-KEM-768 ciphertext, and is the *only* term that differs between the classical and post-quantum framing
ECDSA P-256 CertificateVerify signature	72	measured; counted as certificate material, exactly as the post-quantum rows count their 2,420 B ML-DSA CertificateVerify
server flight framing, applied uniformly	1,354	ciphertext 1,088 + ServerHello 95 + EncryptedExtensions 40 + CertificateRequest 25 + Certificate framing 18 + CertificateVerify header 8 + Finished 36 + records 44

Layers, defined once so no column smuggles the other's bytes. *Raw material* is cryptographic objects only, keys and signatures. *With encoding* adds X.509 and SCT structure. *Server first flight* adds the uniform 1,354. Two of the framing components are estimates (EncryptedExtensions and CertificateRequest, 59 bytes of slack between them at the RFC minimums the floor below uses, or 81 counting the record-layer reduction it also applies); everything else is specified or measured.

PROFILE	RAW MATERIAL	WITH ENCODING	SERVER FIRST FLIGHT	FITS IW10?
classical: ECDSA P-256, ietf.org chain	3,393 measured	+72 CertificateVerify = 3,465	+266 framing +32 X25519 = 3,763	yes, 10,837 to spare
web: ML-DSA-44 leaf, ML-DSA-65 intermediate, 2 SCTs 1,312+3,309+1,952+3,309+2,420+2×2,420	17,142	+660+500+2×53 = 18,408	+1,354 = **19,762**	no, over by 5,162
mesh, ML-DSA-65 intermediate 1,312+3,309+1,952+3,309+2,420	12,302	+660+500 = 13,462	+1,354 = **14,816**	no, over by 216
mesh, ML-DSA-44 throughout 1,312+2,420+1,312+2,420+2,420	9,884	+660+500 = 11,044	+1,354 = **12,398**	yes, 2,202 spare

Why the verdict does not rest on an estimate. The middle row misses by 216 bytes, which is close enough to the framing slack to be worth testing. Zero every estimate to its RFC minimum: 12,302 raw + 1,160 encoding + 1,088 ciphertext + 185 of irreducible framing (ServerHello 95, minimal EncryptedExtensions 6, Certificate framing 18, CertificateVerify header 8, Finished 36, one record 22) gives **14,735**, still above 14,600. That profile fails the initial congestion window under the most favourable accounting available to it. The ML-DSA-44 row's 2.2 kB margin is an order of magnitude beyond the same uncertainty. At a realistic MSS of 1,448 with timestamps the window is 14,480 and the margins become 336 over and 2,082 spare.

Three figures are not used and should not be reintroduced. "17,171" was the constructed total including 29 bytes of subject strings, so it was not raw material; raw is **17,142**. "13,681" for the middle row is not reproducible from any stated constant set and is withdrawn. "11,020" was 8,600-with-encoding plus a raw signature, which mixed two layers in one sum. The mesh chain per side is **8,624**, not 8,600. Reconciling the two chain models on the measured constants rather than a flat 568 per certificate: crypto delta (1,952−1,312)+2×(3,309−2,420) = 2,418, encoding delta 1,160−29 = 1,131, so the gap is **1,287**, not 1,311. Put both models on full encoding and the residual encoding term disappears entirely: **11,042**, the ML-DSA-65-intermediate mesh *chain* per side with encoding, against **8,624**, the all-44 chain per side with encoding, is exactly the crypto delta, 2,418. Neither figure is the 11,044 in the table above, which is the all-44 chain *plus its CertificateVerify* and lands two bytes away by coincidence.

One assumption flagged. The web row assumes ML-DSA-44 SCT signatures. CT logs today sign with ECDSA P-256 at about 72 bytes. With ECDSA SCTs the web flight falls to about 15,066, which still exceeds 14,600 by 466, so that row's verdict survives the assumption too. Also note the pq-crystals site lists Dilithium3's signature as 3,293 where FIPS 204 specifies **3,309** for ML-DSA-65, the commitment hash having grown from 32 to 48 bytes between the round-3 submission and the standard.

1.1c Measured against fourteen public endpoints, 27 August 2026

The classical row above rests on one host measured once. This is the range it sits in. Two connections per host, both to the same pinned address and the same SNI, one to mint a session ticket and one to offer it back. Certificate handshake only: no credential is sent, and a ticket is never offered to a host that did not issue it. Harness, panel and per-host JSON are public in `github.com/quantakrypto/tls-resumption-tests`, directory `public-endpoints/`, so every figure here can be recomputed from the run files rather than taken on trust. The key-exchange and resumption columns are also measured daily over a larger panel by `quantakrypto/pqc-observatory`, which is public, so the two columns that carry the most weight here can be checked against a second instrument by anyone.

HOST	KEY EXCHANGE GROUP	CERTS	CHAIN B	TICKET LIFE S	RESUMED
aws.amazon.com	X25519MLKEM768	3	3,916	72108	yes
docs.aws.amazon.com	X25519MLKEM768	3	3,778	50488	yes
cloud.google.com	X25519MLKEM768	3	4,812	172800	yes
azure.microsoft.com	X25519MLKEM768	3	5,785	83100	yes
learn.microsoft.com	X25519MLKEM768	3	3,725	83100	yes
www.cloudflare.com	X25519MLKEM768	3	2,493	64800	yes
developers.cloudflare.com	X25519MLKEM768	3	2,515	64800	yes
www.fastly.com	X25519MLKEM768	2	2,530	86400	yes
docs.fastly.com	X25519MLKEM768	2	2,544	86400	yes
ietf.org	X25519MLKEM768	3	2,465	64800	yes
letsencrypt.org	not reported	4	3,576	604800	yes
github.com	not reported	3	2,719	7200	no
google.com	X25519MLKEM768	3	4,812	172800	yes
mozilla.org	not reported	3	3,954	172800	yes

Eleven of fourteen selected the post-quantum hybrid group when it was offered alongside the classical ones, including every provider named in supporting document 2. Fourteen of fourteen issued a session ticket and thirteen resumed. Chain sizes span **2,465 to 5,785 B, median 3,650.5**, so the 3,393 B classical figure is inside the measured range rather than an outlier. Two cautions. `ietf.org` served 3 certificates totalling 2,465 B from a Cloudflare address here, not the 4 certificates totalling 3,393 B recorded on 25 August, so that figure is dated rather than wrong and a range is the sounder basis. And `github.com` issued a ticket and then completed a full handshake when it was offered back, which is a readable negative because the controls confirm the ticket was issued and offerable.

1.1d The same panel, asked and not asked, 28 August 2026

Every ticket figure above was measured on connections that send one `HEAD /`. That condition is not incidental, so it was tested: the same harness and the same panel, run twice back to back, differing only in `--request none` against `--request head`.

HOST	TICKET, SILENT	BANNERS	TICKET, ONE HEAD	BANNERS	RESUMED, SILENT	RESUMED, HEAD
aws.amazon.com	yes	1	yes	1	yes	no
docs.aws.amazon.com	yes	1	yes	1	yes	yes
cloud.google.com	no	0	yes	2	not asked	yes
azure.microsoft.com	yes	2	yes	2	yes	yes
learn.microsoft.com	yes	2	yes	2	yes	yes
www.cloudflare.com	no	0	yes	2	not asked	yes
developers.cloudflare.com	no	0	yes	2	not asked	yes
www.fastly.com	yes	1	yes	1	yes	yes
docs.fastly.com	yes	1	yes	1	yes	yes
ietf.org	no	0	yes	2	not asked	yes
letsencrypt.org	yes	1	yes	1	yes	yes
github.com	yes	2	yes	2	no	no
google.com	no	0	yes	2	not asked	yes
mozilla.org	no	0	yes	2	not asked	yes

A silent connection drew a session ticket from 8 of 14 hosts. One `HEAD /` drew one from all 14. The key exchange did not move between the two days: **eleven of fourteen selected `X25519MLKEM768` on both 28 August runs**, the same eleven as the 27 August run in 1.1c, and every chain was byte-identical in length to the day before. That is what the paper's section 2 refers to when it says the same eleven on a second run. The six that withhold until asked are every Google-fronted and every Cloudflare-fronted name on the panel and nothing else: `cloud.google.com`, `google.com`, `mozilla.org`, `www.cloudflare.com`, `developers.cloudflare.com` and `ietf.org`. It is a property of the terminating stack, not of the site's own policy: neither mozilla.org nor ietf.org is a cloud provider, and both land in that group by whose edge terminates their TLS.

This is why the paper's section 4 records the condition alongside the corroboration. Hebrok et al. probed for session tickets by sending an HTTP request, section 5.1.1, so the two measurements are comparable; a reader who reproduces ours with a bare handshake would measure 8 of 14 and reasonably conclude the paper had overstated how widely resumption is offered.

Two cautions, both recorded rather than smoothed over. The absence of a ticket on a silent connection is bounded by the harness's two-second hold, not absolute; a server that volunteers one after three seconds of silence would be recorded here as withholding. And `aws.amazon.com` is not stable across runs. At one address, 13.32.121.33, it resumed on 27 August and refused on 28 August, with ticket lifetime hints of 72,108 and 56,146 seconds; the silent run resolved to a different address, 143.204.238.22, and resumed there. Each run pinned one address across both its connections, so each result is readable within its run, but the outcome is not a property of the host. **A single refusal from a CDN anycast address is not evidence of a policy.** What distinguishes `github.com` is that its refusal is stable across every run and its lifetime hint never moves. Nothing in the paper rests on a single reading.

1.2 What post-quantum multiplies, and what it does not

The quoted line is Filippo Valsorda's *commit message* on CL 497895, not an issue comment, and it compresses an argument Marten Seemann made in golang/go#31641 in 2019. Credit the wording to Valsorda and the idea to Seemann.

The cost runs the opposite way to the intuitive reading. CL 497895 did not merely decline to re-verify; it *serialised the verified chains into the ticket*, in its own words "a tradeoff making the ticket bigger to save computation". Go 1.21's release notes say the same: "This makes session tickets larger when client certificates are in use." One refinement: the peer certificates were already serialised before that change, so what CL 497895 added was moving `verified_chains` out of the client-only branch, at a marginal cost of the intermediates and root per chain, the leaf elided. So Go already pays the ticket-size cost. Measured against

that status quo, the check Go declined costs only re-verification CPU, which ML-DSA does not raise. Post-quantum therefore inflates a price Go is **already** paying and leaves the price of the declined check roughly flat, which **weakens** Go's stated rationale rather than strengthening it.

That pressure is already live without any post-quantum algorithm. In January 2026 Valsorda opened golang/go#77379, "allow omitting certificates from session ticket", paraphrasing Seemann that carrying them "makes TLS 1.3 resumption nearly useless, because it doesn't save round-trips and the client sends a ticket which is as large as the original chain (or larger...)". Post-quantum multiplies exactly that ticket. Two things to keep straight. What Go declines is full re-verification, not every check: CL 737700 (January 2026) added a deliberately cheap chain-validity and root-in-pool check, and `VerifyPeerCertificate` remains documented as "not invoked on resumed connections". And none of golang/go#31641, #77217 or CL 497895 mentions post-quantum at all: **the extrapolation is this paper's, not the Go team's**.

On CPU, stated conservatively: ML-DSA verification is in the same order as ECDSA P-256. Two data points, both sourced, because they disagree in *direction* and the disagreement is the point.

Two different ML-DSA-44 verification figures appear in these documents and they are not in conflict. eBACS/SUPERCOP measures a median of 69,059 cycles for Dilithium2, the pre-standardisation ML-DSA-44, on an AMD Ryzen 7 7700 (Zen 4, AVX2); the pq-crystals team's own table reports 118,412 cycles on an Intel Core i7-6600U (Skylake, AVX2). Same operation, different hardware and different suites. Two cautions: eBACS labels the scheme `dilithium2` rather than `mldsa44`, and its signing row on that machine is flagged as unstable, so only its verify and keygen figures are cited here. Where a calculation needs a signature and a KEM cost together, both come from eBACS on the same machine (ML-KEM-768 keygen 26,255, encapsulation 25,929, decapsulation 27,069), because the pq-crystals Dilithium and Kyber tables were measured on Skylake and Haswell respectively and must not be added together.

SOURCE	ML-DSA-44 VERIFY	ECDSA P-256 VERIFY	RATIO
SUPERCOP/eBACS `supercop-20260627`, machine `hertz` (AMD Ryzen 7 7700, Zen 4); ML-DSA via `dilithium2` **avx2**, ECDSA via `ecdonaldp256` **opensslnew** (reported as the `ecp_nistz256` assembly path; the archived table lists medians, not implementation names)	69,059 median cycles	235,984 median cycles	**ML-DSA 3.4× faster**
Local run, `openssl speed -seconds 3`, OpenSSL 3.6.1, Apple M1 Pro (arm64), 26 Aug 2026	10,359 verify/s	14,567 verify/s	**ML-DSA 1.4× slower**

The two are not in conflict; they measure different builds. Back-of-envelope on the local run puts ML-DSA-44 verification near the *reference-C* cycle count, which is the inference that OpenSSL 3.6 ships unvectorised ML-DSA against hand-written P-256 assembly. That is an inference from the cycle count, not documented fact. Note also that the local figure is arm64, so it supports the implementation-parity point and should not be quoted as an x86-64 number. Because implementation parity decides the sign of the comparison, **the paper rests only on the weaker claim that verification cost is roughly flat**, which holds under either row. ML-DSA-65 verification on the same eBACS machine is 113,813 cycles, still below P-256.

1.3 Merkle Tree Certificates

Cited from **draft-ietf-plants-merkle-tree-certs-05, 6 July 2026**. Section 7.2: "This procedure only replaces the signature verification portion of X.509 path validation. The relying party MUST continue to perform other checks, such as checking expiry." The draft's section 6.5 computes its size case at a 7-day certificate lifetime; the draft sets no normative maximum lifetime.

A correction worth recording. Earlier drafts of this paper quoted "certificate expiration replaces an external revocation signal like CRLs or OCSP" and a 14-day recommendation, and attributed them to the working-group draft. That text is real but belongs to the *individual* draft, draft-davidben-tls-merkle-tree-certs section 5.1, present through revision -04 and removed at -05. The working-group draft says close to the opposite at section 12.7: "This document does not define a new certificate-level revocation mechanism. Existing mechanisms like CRLs and OCSP apply unchanged to Merkle Tree certificates."

1.4 AuthKEM

draft-celi-wiggers-tls-authkem-07, 4 May 2026, authors Wiggers, Celi, Schwabe, Stebila, Sullivan: the handshake keeps the same messages, adds a KEMEncapsulation message, and "does not use the CertificateVerify one". (The draft's own sentence contains a typo at this point, so only the unambiguous clause is quoted here and the rest is paraphrased.) Authentication is by KEM decapsulation against the certified key, mixed into the key schedule.

2. References

This is the **shared bibliography for the whole submission**, reproduced identically in each supporting document so that any one of them can be read on its own. Not every entry is cited in this particular document.

S. Hebrok, T. L. Storm, F. M. Cramer, M. Radoy, J. Somorovsky, "STEK Sharing is Not Caring: Bypassing TLS Authentication in Web Servers using Session Tickets", 34th USENIX Security Symposium, August 2025. https://www.usenix.org/conference/usenixsecurity25/presentation/hebrok

RFC 9846, TLS 1.3, section 4.7.1, July 2026. https://www.rfc-editor.org/rfc/rfc9846.txt

RFC 9190, EAP-TLS 1.3. https://datatracker.ietf.org/doc/rfc9190/

L. Chuat et al., "SoK: Delegation and Revocation, the Missing Links in the Web's Chain of Trust", IEEE EuroS&P 2020. https://arxiv.org/abs/1906.10775

R. Holz et al., "The Era of TLS 1.3", IMC 2019. https://arxiv.org/abs/1907.12762

Cloudflare, "Resolving a Mutual TLS session resumption vulnerability", 7 February 2025. https://blog.cloudflare.com/resolving-a-mutual-tls-session-resumption-vulnerability/

BoringSSL include/openssl/ssl.h; Chromium net/socket/ssl_client_socket_impl.cc; NSS lib/ssl/tls13con.c. All retrieved 25 August 2026.

nginx changeset 8086:496241338da5, 12 October 2022, shipped in 1.23.2. HAProxy commit 9f1d590999c2, 18 August 2026.

`Scripts: pdf/rows.py (the table and its derived counts), fftest/server.py, fftest/nonbrowser.sh, fftest/RESULT.md.`