

Layered Performance Analysis of TLS 1.3 Handshakes

Traditional, Hybrid PQ, and Pure PQ Key Exchange

David Gómez-Cambronero

Telefónica, Madrid, Spain

Daniel Munteanu

Keysight Technologies, Bucharest, Romania

Ana Isabel González-Tablas

Universidad Carlos III de Madrid, Spain



Agenda

- 1 Motivation: the quantum threat
- 2 PQC standardization & migration timelines
- 3 Gap in prior work & our contributions
- 4 Methodology: the 5-layer decomposition
- 5 Laboratory architecture & setup
- 6 Results — latency per protocol layer
- 7 Normalized overhead & payload effects
- 8 CPU / network & end-to-end perspective
- 9 Conclusions & future work
- 10 Q&A

The quantum threat is a migration problem

- Cryptographically relevant quantum computers threaten today's public-key crypto (ECDH, RSA).
- “Store now, decrypt later”: adversaries harvest encrypted traffic today to decrypt once quantum-capable.
- NIST finalized the first PQC standards in August 2024:
 - FIPS 203 — ML-KEM (key encapsulation)
 - FIPS 204 — ML-DSA (signatures)
 - FIPS 205 — SLH-DSA (hash-based signatures)
- Migration is inevitable — but what is the real performance cost before production?

The question

How much latency does PQC actually add and exactly where in the TLS 1.3 transaction does it hit?

Standardization & migration timelines

- NIST standards adopted by Open Quantum Safe (liboqs) and OpenSSL; network vendors now integrating.
- NSA CNSA 2.0 timelines for national security systems:
 - ML-KEM for key establishment by 2030
 - Quantum-resistant algorithms exclusively by 2033
- Parallel initiatives across the EU and other jurisdictions.
- Result: a pressing need for empirical performance data to inform migration planning.

2024

First NIST PQC standards finalized

2030

CNSA2.0:ML-KEM key establishment

2033

Quantum-resistant algorithms only

The gap in prior work & our contributions

The gap

Prior studies report only aggregate TLS handshake latency — not where the cost originates.

100 TPS

sustained, 5 min each

>30

performance tests

Our contributions

- 5-layer latency decomposition of TLS 1.3 connections.
- Traditional vs. hybrid vs. pure PQC, side by side, under realistic load.
- Finding: the handshake exchange is algorithm-neutral.
- A public data-reduction tool for per-layer stats from pcaps.

~1M

total requests

5

protocol layers analyzed

Decomposing the HTTP-over-TLS transaction

One TLS 1.3 (1-RTT full) connection → five measurable phases. The two intermediate layers are where PQC cost concentrates.



Attribution, not aggregation: measuring each phase separately lets us pinpoint the exact phase where PQC overhead originates, rather than observing only the end-to-end effect. Scope: 1-RTT full handshake (no 0-RTT, no session resumption).

Why five layers? What each one tells us

TCP Handshake

SYN → SYN-ACK

Control layer, must be algorithm-independent; variation signals system load.

TCP → TLS delay

SYN-ACK → ClientHello

Client-side TLS init & ephemeral key-material generation, where PQC hits hardest.

TLS Handshake

ClientHello → Finished

On-the-wire key exchange, certificate transmission and verification.

TLS → App delay

Finished → HTTP GET

Residual overhead after the secure channel is established.

Application

GET → 200 OK

Client-to-backend response time, should be algorithm-independent.

Laboratory / test-bed architecture



KEx group and response body size are controlled independently by CyPerf. Captures + TLS key logs feed the public data-reduction script.

Environment configuration & reproducibility

Server side — Nginx reverse proxy

- Nginx 1.27.3, TLS-terminating endpoint
- OpenSSL 3.4.0 + liboqs 0.12.0 + OQS provider 0.8.0
- Alpine 3.21 base image; RSA-2048 server certificates

Client side — Keysight CyPerf

- High-scale stateful HTTPS traffic generator
- Native ML-KEM key-exchange support
- 100 TPS sustained (well below tool capacity)

Reproducibility & monitoring

- Containerized: pinned Docker images & versions
- Public Git repository with full setup
- Signature held constant (RSA-2048) → differences attributable to key exchange only
- Metrics via Prometheus + Grafana + CyPerf
- Per-test SSLKEYLOGFILE export enables packet decryption & correlation

Key-exchange configurations & test scenarios

Traditional	Hybrid	Pure post-quantum
x25519	x25519_MLKEM512 x25519_MLKEM768	MLKEM512 MLKEM1024

Backend response body scenarios

Direct minimal body	4 KB reference payload	40 KB large payload
-------------------------------	----------------------------------	-------------------------------

Metrics & statistical analysis

Per-layer latency

mean, p50/p90/p95/p99, min/max, SD; plus true per-connection E2E (SYN → 200 OK).

Size & key material

ClientHello / ServerHello length; key_share payload bytes per group.

Resource utilization

client & server CPU (system/user/idle); network throughput (Mb/s).

Effect-size criterion — Glass's Δ

$\Delta = (\text{mean_PQC} - \text{mean_x25519}) / \text{SD_x25519}$ — differences judged against the baseline's own measurement variability.

$|\Delta| < 0.2 \rightarrow$ negligible

$0.2 - 0.8 \rightarrow$ small-medium

$|\Delta| > 0.8 \rightarrow$ large

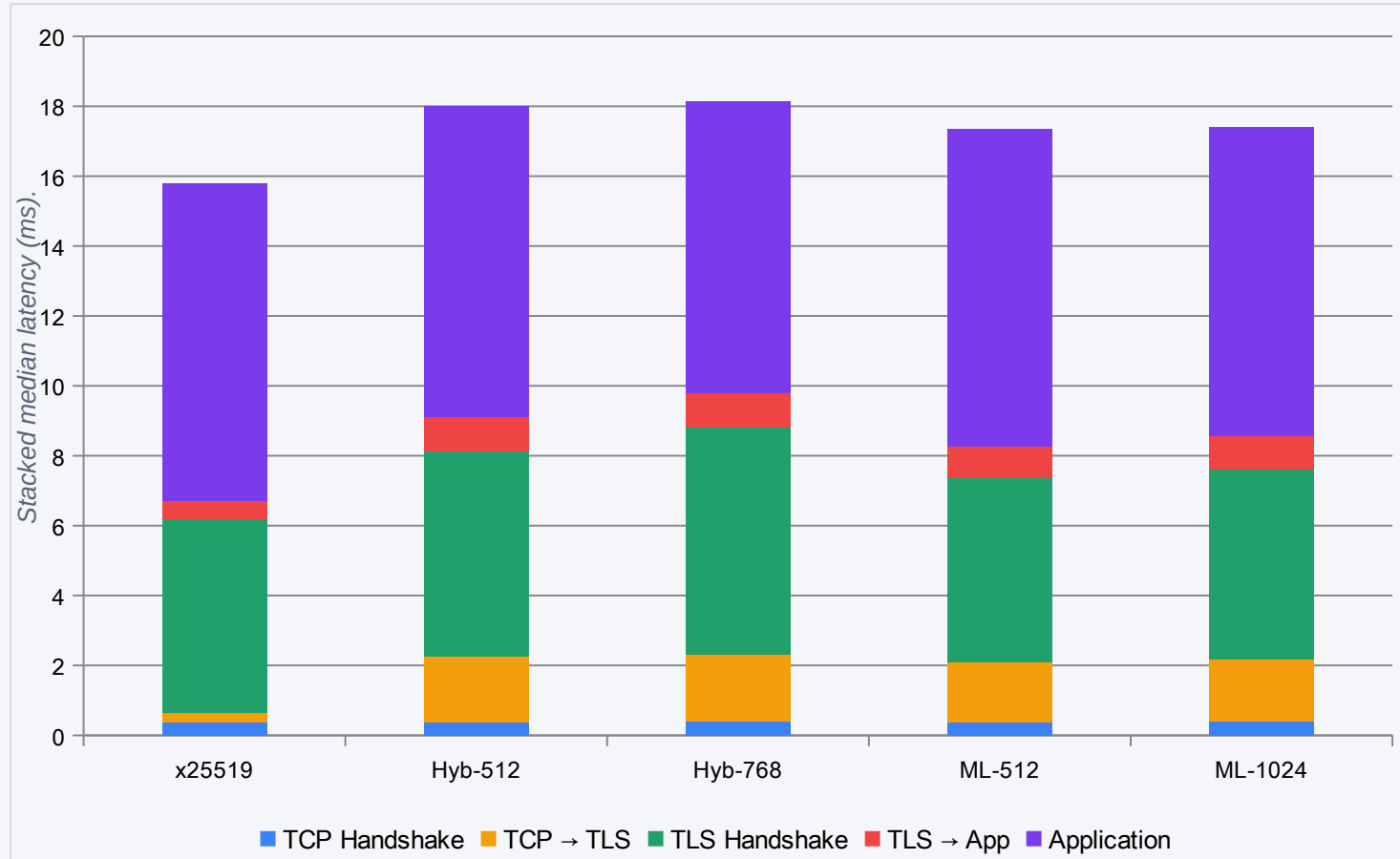
Full table

Table 2. Latency percentiles and standard deviation per protocol layer. Rows labeled (4 KB) and (40 KB) refer to the backend response body size. Comparing x25519 (classical), x25519_MLKEM512 (hybrid), x25519_MLKEM768 (hybrid), MLKEM512 (pure PQC), and MLKEM1024 (pure PQC). The 40 KB scenario is included for x25519 and x25519_MLKEM768 to support the analysis in Section 4.7.

Layer (Timestamp Start → End)	p50 (ms)	p95 (ms)	p99 (ms)	SD (ms)
<i>TCP handshake (SYN → SYN-ACK)</i>				
x25519 (4 KB)	0.360	0.694	0.957	0.248
x25519_MLKEM512 (4 KB)	0.390	3.060	4.147	0.999
x25519_MLKEM768 (4 KB)	0.402	2.720	4.139	1.016
MLKEM512 (4 KB)	0.381	2.882	3.892	0.969
MLKEM1024 (4 KB)	0.393	2.397	3.901	0.915
x25519 (40 KB)	0.401	0.781	1.251	16.993
x25519_MLKEM768 (40 KB)	0.417	3.386	5.048	10.163
<i>TCP-to-TLS delay (SYN-ACK → ClientHello)</i>				
x25519 (4 KB)	0.294	0.635	0.800	0.194
x25519_MLKEM512 (4 KB)	1.866	4.122	5.664	1.196
x25519_MLKEM768 (4 KB)	1.903	3.900	5.534	1.411
MLKEM512 (4 KB)	1.726	3.577	5.043	1.065
MLKEM1024 (4 KB)	1.772	3.608	5.088	1.044
x25519 (40 KB)	0.289	0.702	0.901	0.232
x25519_MLKEM768 (40 KB)	1.915	4.097	5.598	1.421
<i>TLS handshake (ClientHello → Finished)</i>				
x25519 (4 KB)	5.547	10.903	12.697	2.893
x25519_MLKEM512 (4 KB)	5.879	11.296	13.256	2.871
x25519_MLKEM768 (4 KB)	6.495	11.692	13.650	2.982
MLKEM512 (4 KB)	5.253	9.999	12.010	3.798
MLKEM1024 (4 KB)	5.450	10.578	12.209	2.701
x25519 (40 KB)	5.791	10.886	12.242	2.827
x25519_MLKEM768 (40 KB)	6.105	10.715	12.603	7.055
<i>TLS-to-Application delay (Finished → HTTP GET)</i>				

RESULTS

Latency breakdown by protocol layer (4 KB, p50)



- TLS handshake and application dominate the total and stay flat across algorithms.
- PQC adds a visible slice only in the TCP → TLS layer.
- tail (p95), TCP and TCP → TLS grow disproportionately under PQC; TLS and app scale proportionally

Note

Bars stack the per-layer p50 (p95) values; since percentiles are not additive, the stacked height is a lower bound and does not equal the true E2E percentile reported in slide 20 (paper Section~4.9).

RESULTS

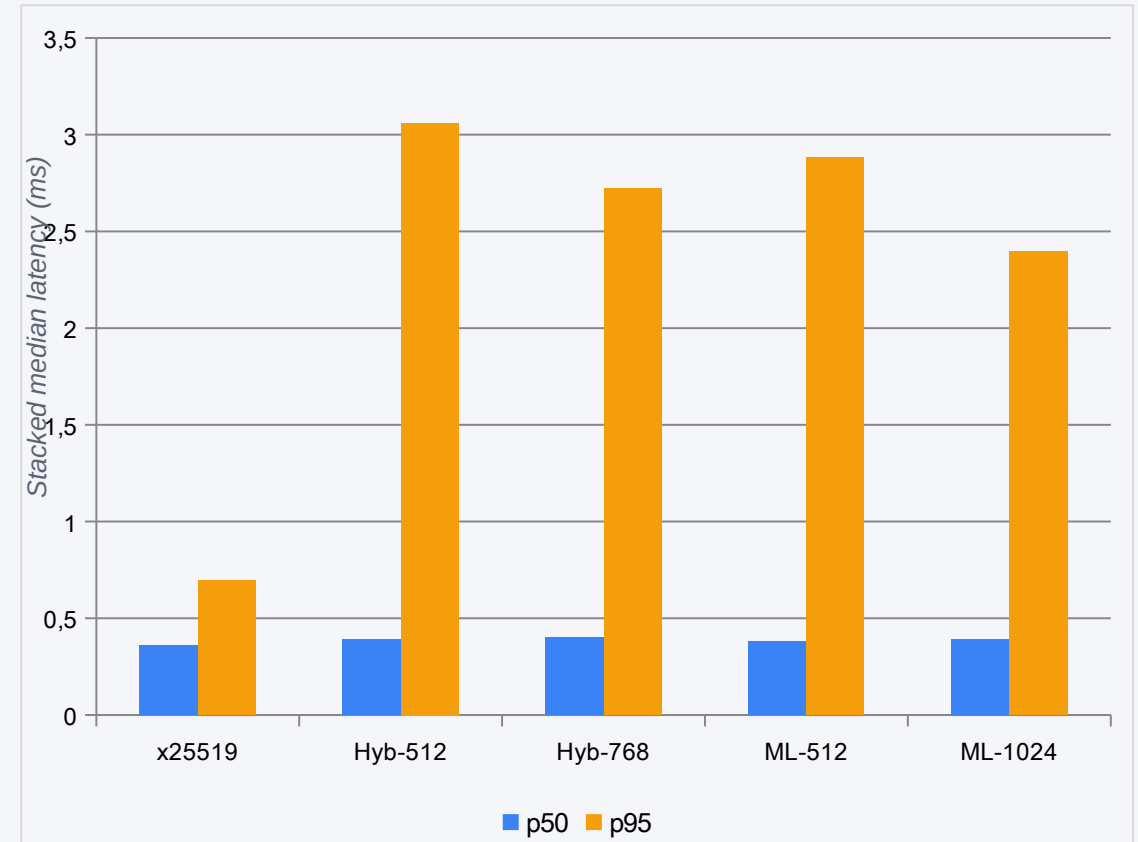
TCP handshake: algorithm-independent

Median range across ALL configs

0.360 – 0.402 ms

max difference 0.042 ms $\rightarrow \Delta = 0.17$ (negligible)

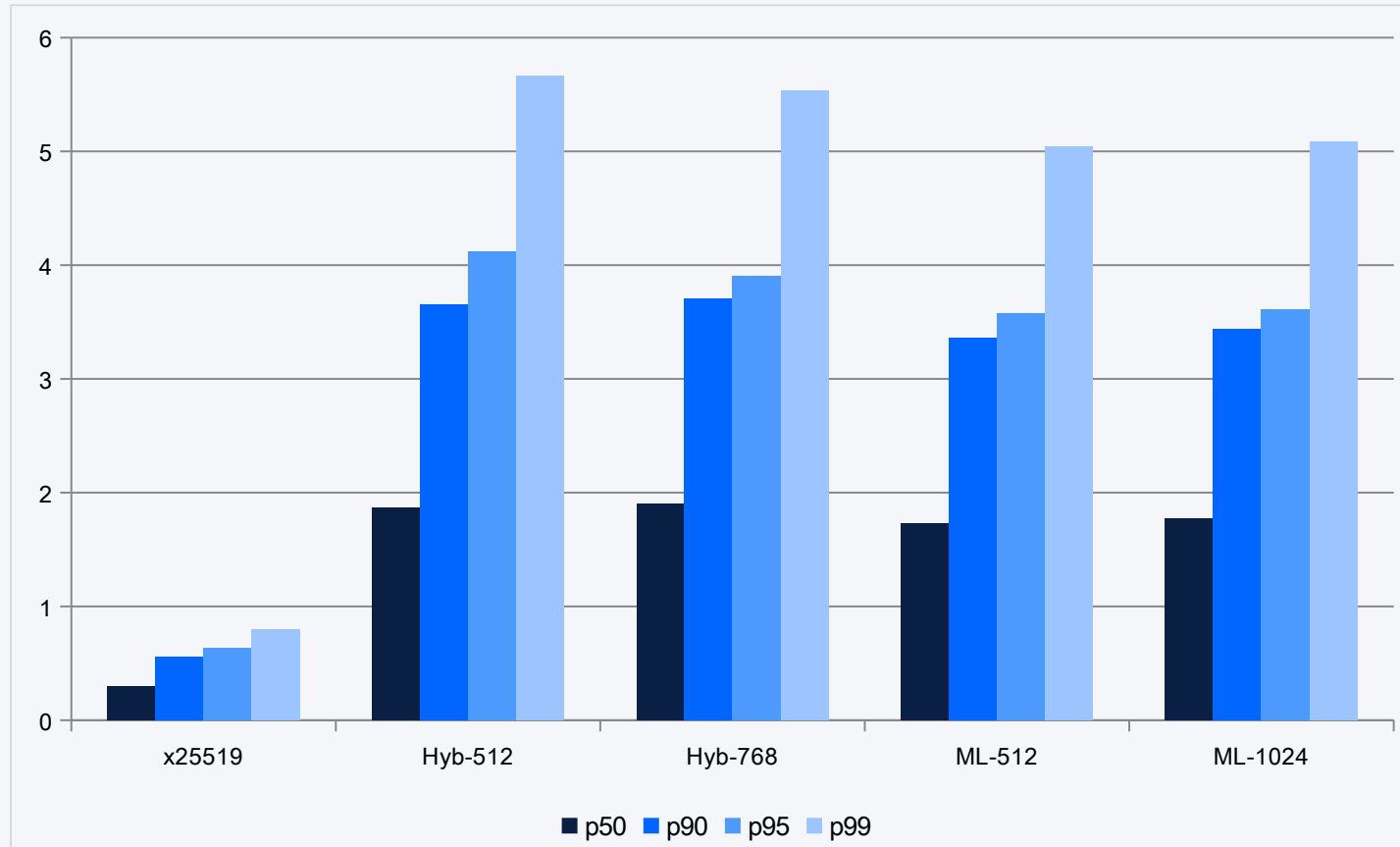
- SYN $\rightarrow$ SYN-ACK genuinely independent of the cryptographic algorithm.
- Confirms PQC overhead does NOT propagate to lower protocol layers.
- p95 tail rises under PQC, but this is system load, since TCP precedes any key-exchange computation.



TCP handshake latency (ms): median vs. tail.

RESULTS

TCP → TLS delay: the dominant PQC cost



TCP → TLS delay (SYN-ACK → ClientHello), full percentile profile (ms, 4 KB).

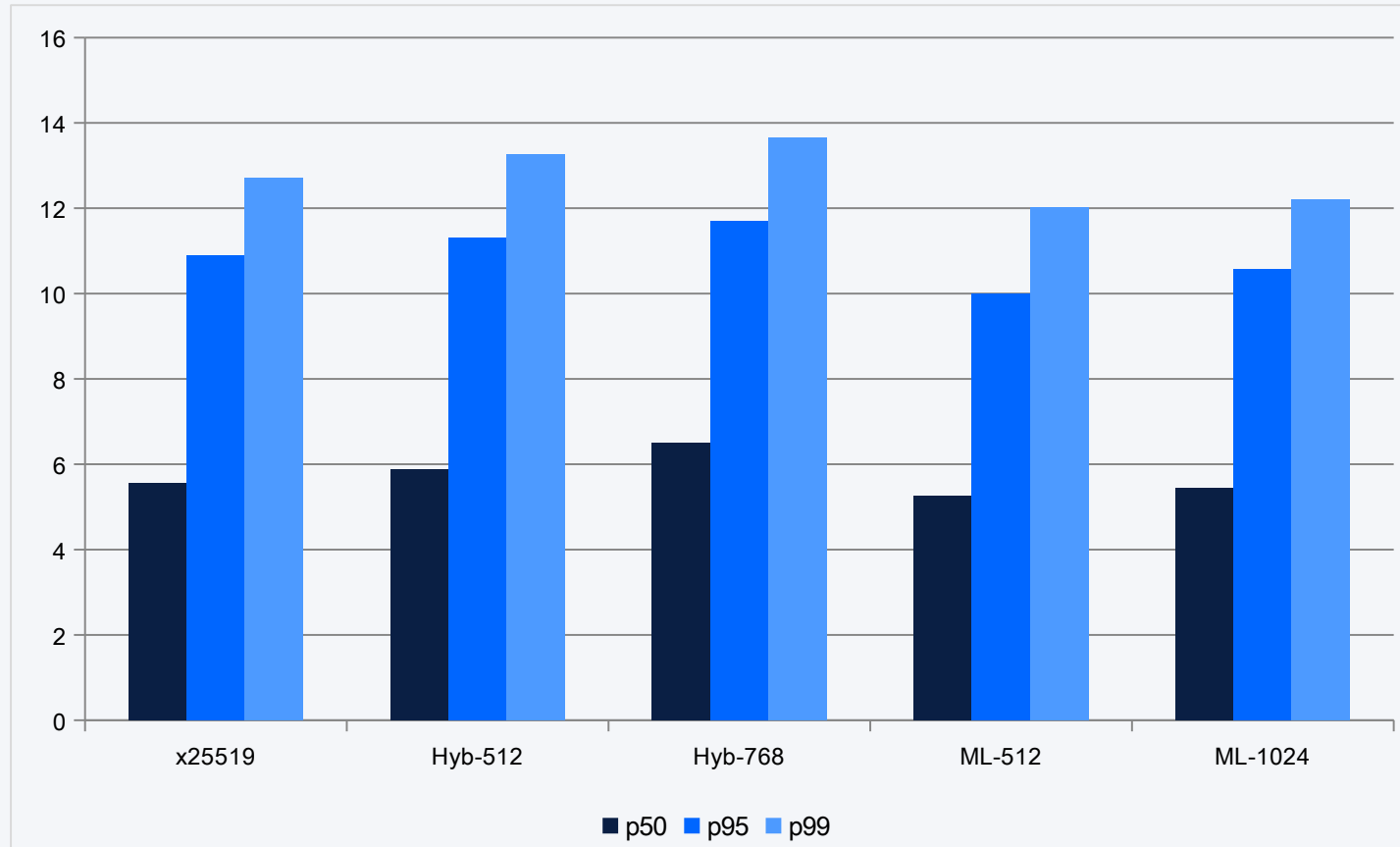
~6× slowdown

0.29 ms → 1.73–1.90 ms ($\Delta \approx 7.7$, large)

- Cost = client-side key-material generation + ClientHello construction.
- Consistent across all percentiles → systematic, not outliers.
- Independent of backend payload size.
- Optimizable: precompute ephemeral key shares off the critical path.

RESULTS

TLS handshake exchange: algorithm-neutral



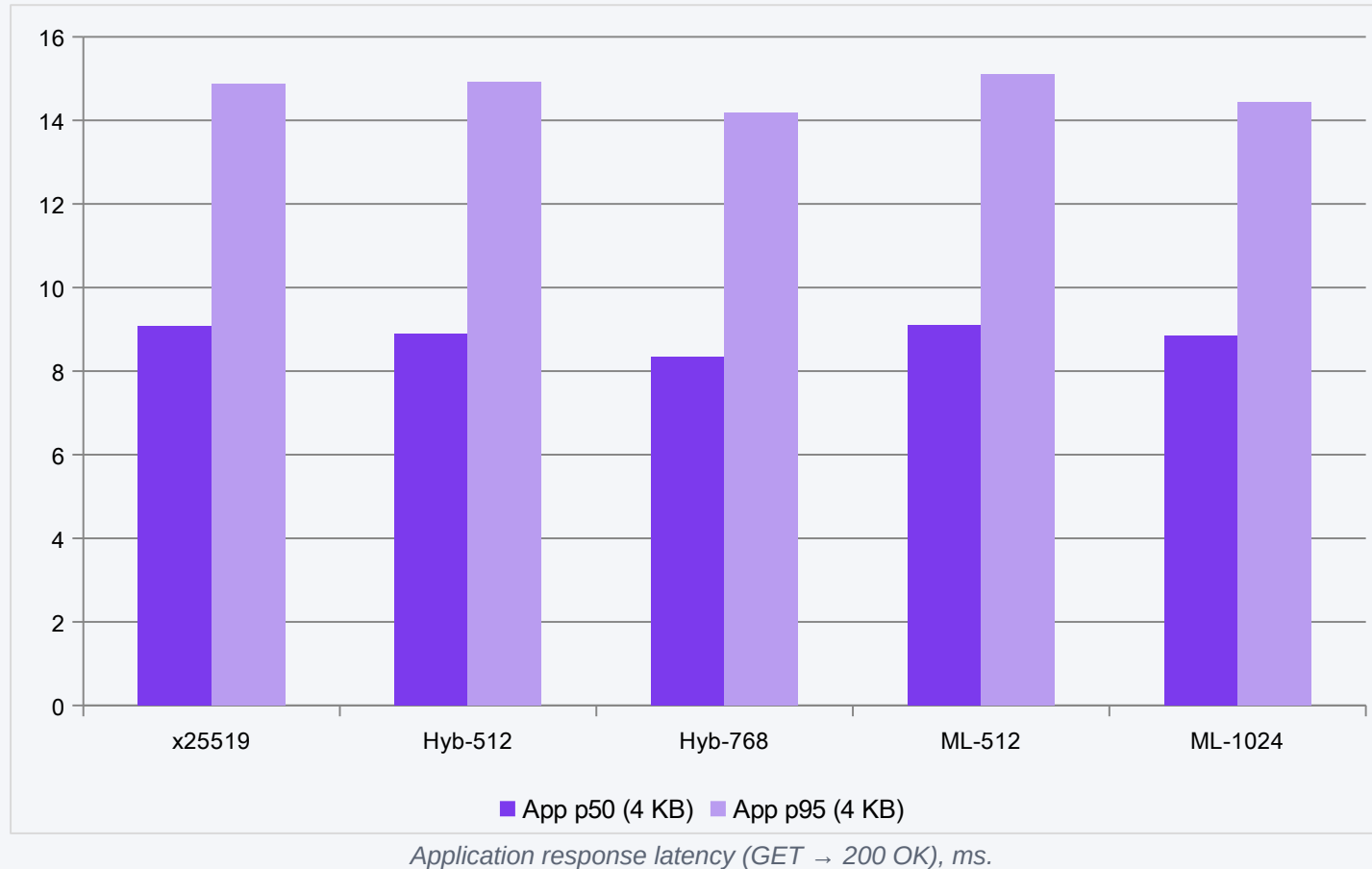
TLS handshake latency (ClientHello → Finished), ms, 4 KB.

$$\Delta = 0.03 - 0.33$$

negligible to small — within noise

- All configs fall within 5.25 – 6.50 ms at the median.
- Pure ML-KEM nominally faster, but indistinguishable from baseline.
- Key insight: neutrality applies to the on-the-wire exchange — NOT to ClientHello construction (measured in TCP → TLS).

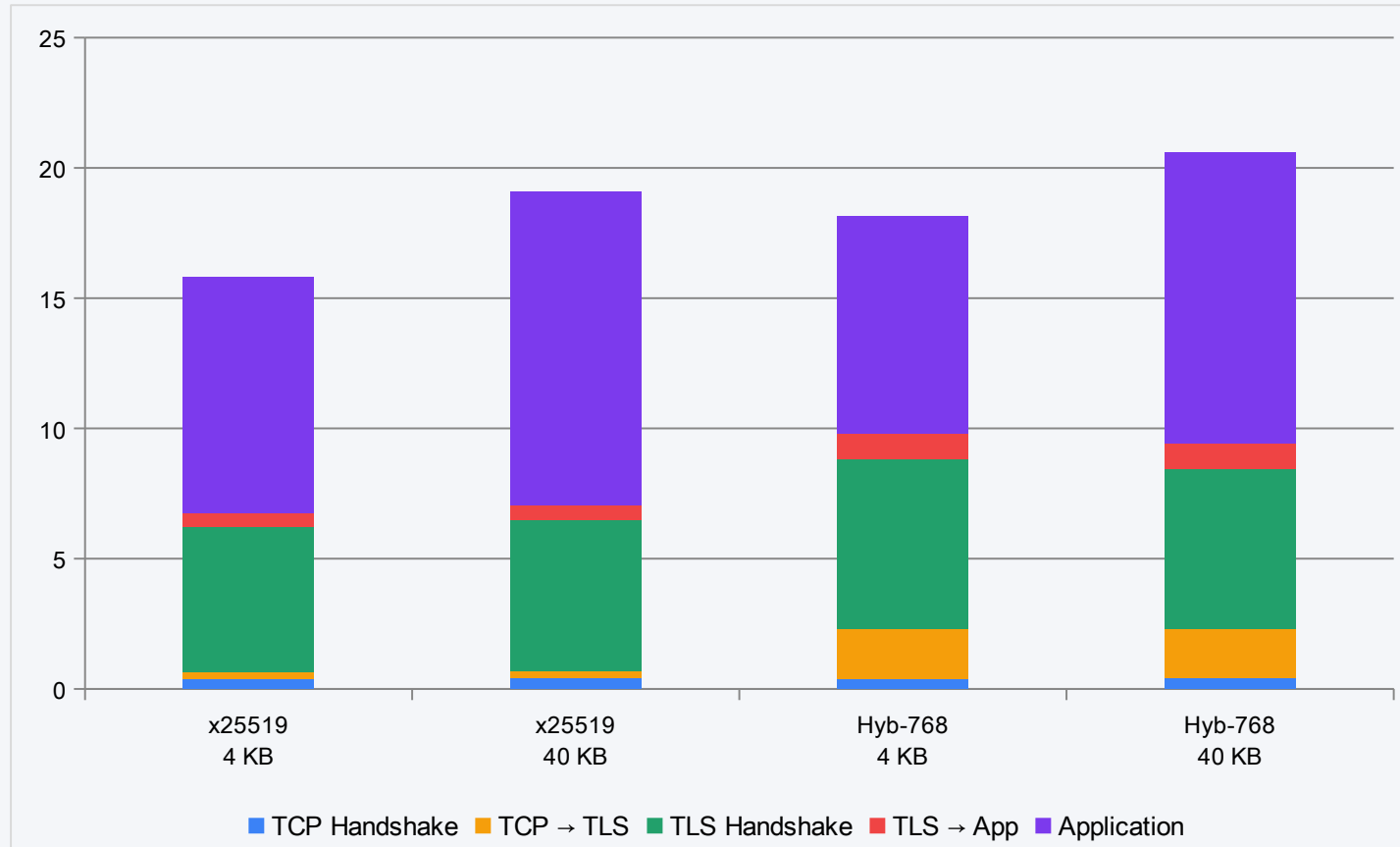
Post-handshake & application: negligible impact



- TLS → App: small absolute effect (0.37 – 0.48 ms), stable across configs.
- Application response dominated by network path + payload (~9 ms @ 4 KB).
- Max inter-algorithm variation 0.74 ms → $\Delta = 0.21$ (negligible).
- PQC does NOT degrade user-perceived application performance.

RESULTS

Payload size dilutes the relative PQC overhead

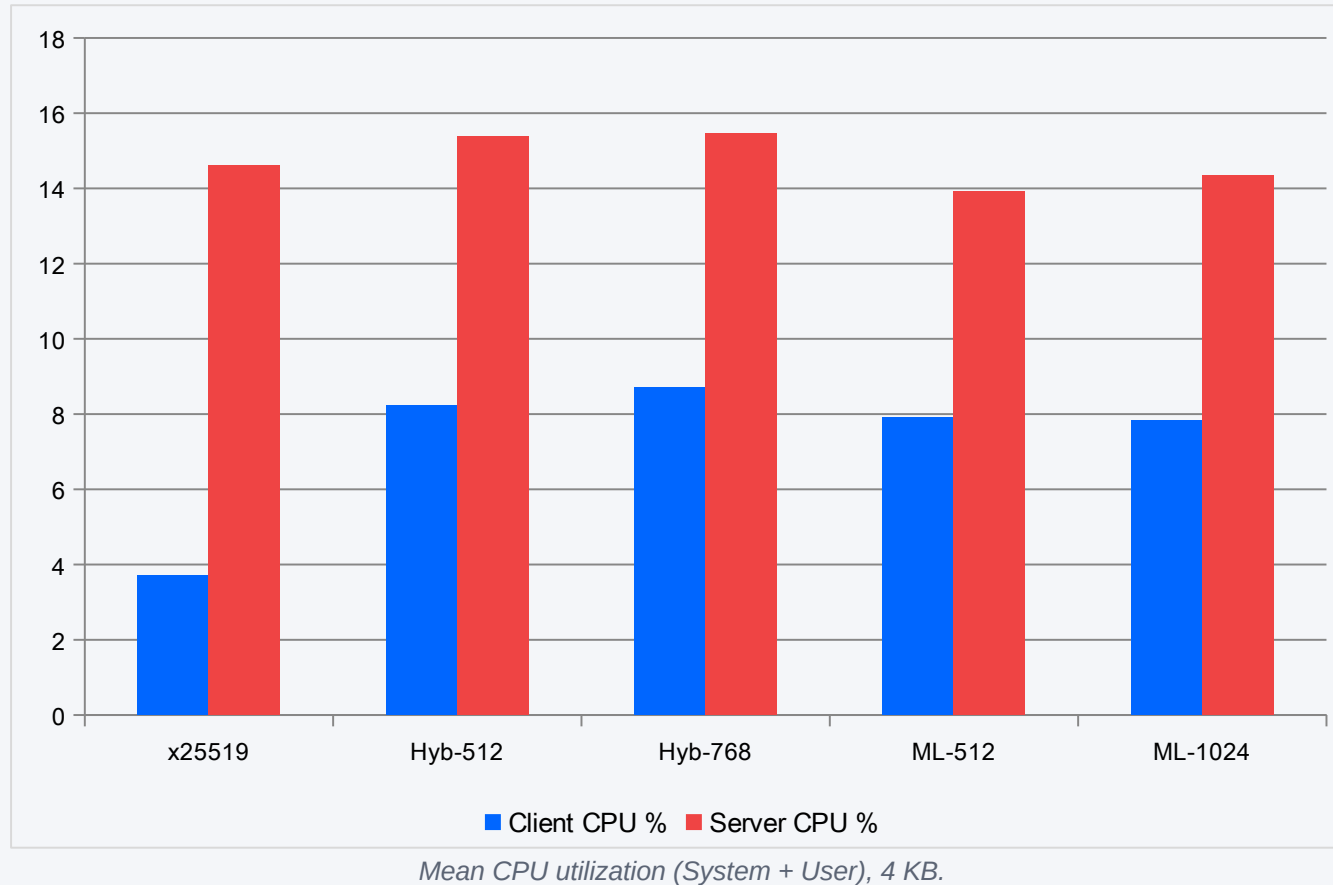


End-to-end decomposition, 4 KB vs. 40 KB, p50 (ms).

- Crypto layers unchanged from 4 KB to 40 KB; only the application layer grows.
- Absolute PQC overhead stays constant (~2.3 ms).
- Relative overhead drops: ~15% (4 KB) → ~8% (40 KB) at the median.
- PQC cost is front-loaded at connection establishment.

RESULTS

CPU / network usage & the implication of MiTM(man-in-the-middle) devices



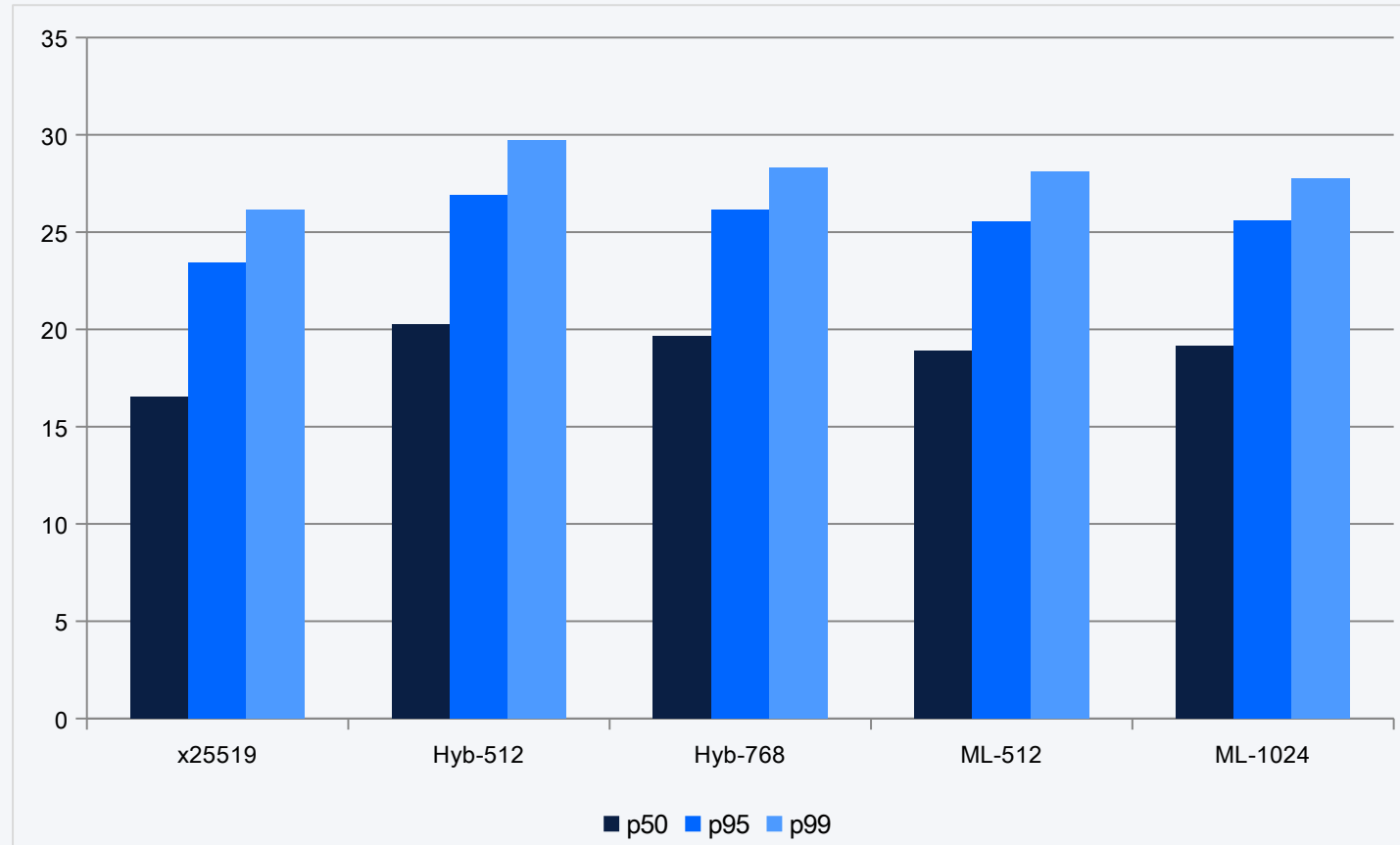
- Client CPU roughly doubles under PQC (3.7% → 8.2–8.7%).
- Server CPU stays 14–15% (hybrids up to +6% relative).
- Traffic 5.2 → 6.6 Mb/s; key_share 32 B → 1568 B.

MiTM inspection devices

Decrypt + inspect + re-encrypt combines client-side and server-side PQC cost at high session rates, the highest expected impact. Test at network level under production load.

RESULTS

End-to-end perspective: a fixed connection cost



End-to-end latency (SYN → 200 OK), 4 KB (ms).

- x25519 baseline: 16.54 ms median.
- Hybrids +19–23%; pure ML-KEM +14–16%.
- PQC = a FIXED ~2–4 ms cost at connection establishment, independent of payload.
- Its relative weight shrinks as payload (and total time) grows.

Conclusions & future work

Conclusions

- PQC overhead concentrates in ClientHello construction (TCP → TLS, ~6×).
- The handshake exchange is algorithm-neutral.
- Application-layer impact is negligible.
- Crypto Overhead Share stays moderate (< 15%).
- Overhead is a fixed connection-setup cost, diluted by larger payloads.

Future work

- Real networks: commercial load balancers & MiTM inspection devices.
- Stress tests beyond 100 TPS to find the breaking point.
- PQ signatures: ML-DSA, Falcon, SPHINCS+.
- More KEM groups & per-primitive profiling.

Already in progress

Native-PQC stacks: Nginx + OpenSSL 3.5, and open-source load injection with Locust (OpenSSL 3.5).

Questions & Discussion

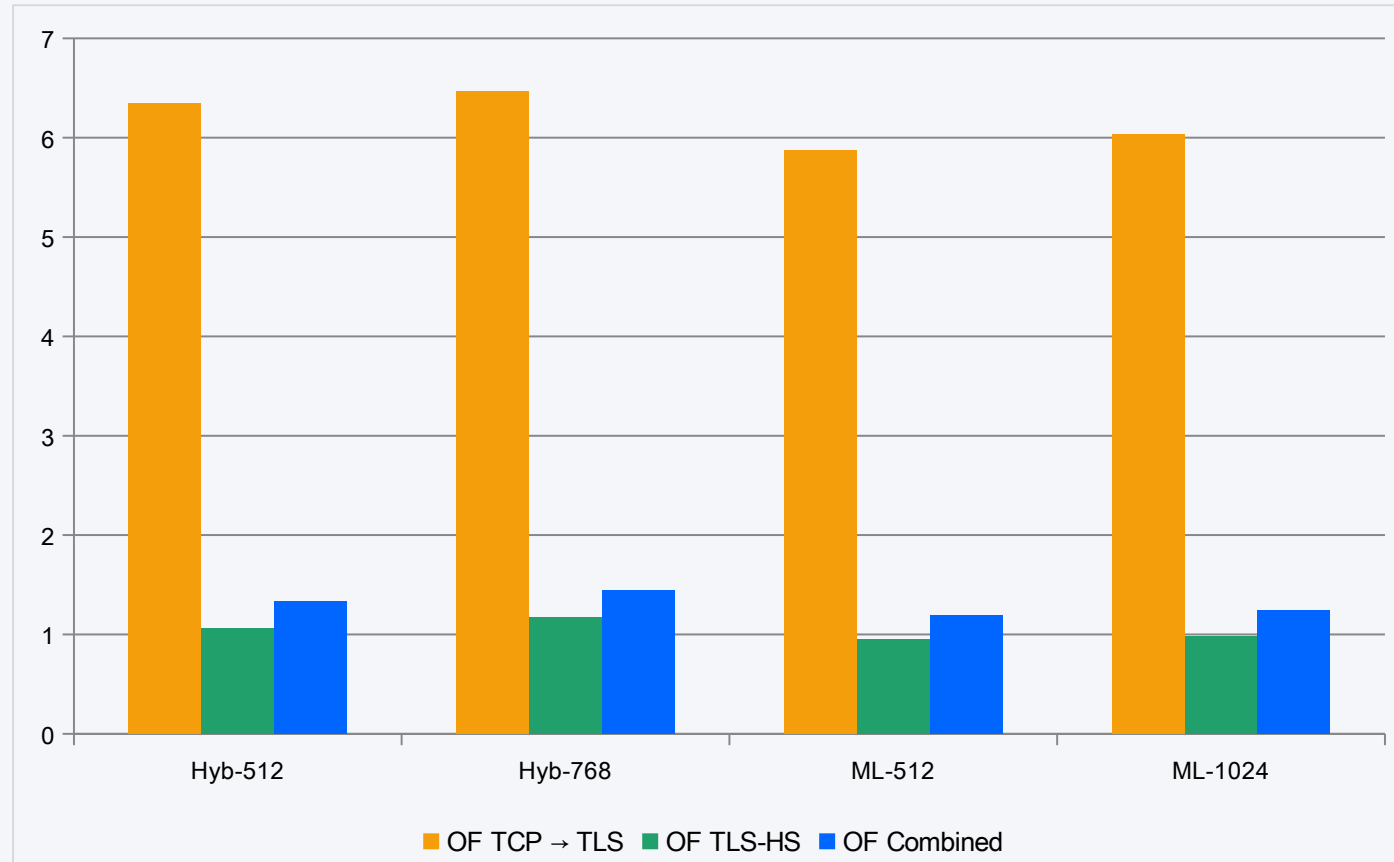
Thank you — David Gómez-Cambroneró · Telefónica

david.gomezcambrenero@telefonica.com

Repository: `github.com/Telefonica/Public_Performance_Quantum_Safe`

RESULTS

Normalized overhead: Overhead Factor & COS



Overhead Factor (L_{PQC} / L_{x25519}) at p50. Baseline OF = 1.0.

Cryptographic Overhead Share

6 – 14% of E2E time

- TCP → TLS: OF $\approx 6\times$ (large, $\Delta \approx 7.7$).
- TLS handshake: OF ≈ 1.0 (negligible).
- Hybrids highest COS (10.6 – 14.1%); pure ML-KEM 6.0 – 7.9%.
- Overall crypto penalty is moderate: $< 15\%$.