

# A Post-Quantum Cryptography Recommendation System with TLS Validation for Heterogeneous Environments

Min-Ho Song<sup>1</sup>, Si-Woo Eum<sup>1</sup>, Seung-Won Lee<sup>2</sup>, Ha-Gyeong Kim<sup>2</sup>, Yu-Lim Hyoung<sup>2</sup>, and Hwa-Jeong Seo<sup>2</sup>

<sup>1</sup> Department of Information Computer Engineering, Hansung University, Seoul 02876, Republic of Korea

smino0906@gmail.com, shuraatum@gmail.com

<sup>2</sup> Department of Convergence Security, Hansung University, Seoul 02876, Republic of Korea

dkajdfhd1@gmail.com, kimhaha4420@gmail.com, yulim4hyoung@gmail.com, hwajeong84@gmail.com

**Abstract.** The choice of a post-quantum cryptography (PQC) algorithm for a concrete deployment is governed by device performance, network conditions, and the practical constraints of TLS stacks, factors that algorithm-centric guidance does not capture. This paper presents a recommendation system that ranks key-encapsulation and signature candidates using benchmarks measured on the target machines and a physically grounded cost model, and confirms as recommended only those candidates that pass end-to-end TLS handshake validation. In experiments on three hardware placements composed of macOS, Linux, and Raspberry Pi machines, the recommended signature algorithm at NIST security level 3 reverses with the server hardware. Handshakes whose server flight exceeds the initial congestion window incur an additional 124ms on a 160ms link, far more than the 44ms attributable to the extra bytes alone. The validation gate further exposes deployment obstacles invisible to microbenchmarks: a standardized signature scheme whose certificates are issued but rejected by the TLS stack, and a wire-format incompatibility between generations of the same crypto provider.

**Keywords:** Post-quantum cryptography · algorithm selection · TLS handshake · cost model · benchmarking · heterogeneous computing.

## 1 Introduction

On a sufficiently large quantum computer, Shor’s algorithm solves, in polynomial time, the integer factorization and discrete logarithm problems on which today’s public-key infrastructure relies [35]. This threat cannot be deferred to the future. An adversary can record encrypted traffic today and decrypt it once a cryptanalytically relevant quantum computer exists, the so-called “harvest now, decrypt later” attack. Key exchange for communications requiring long-term confidentiality must therefore migrate to post-quantum cryptography (PQC) before

such a computer materializes [22]. Accordingly, NIST finalized ML-KEM [25] for key encapsulation and ML-DSA [24] and SLH-DSA [26] for digital signatures as federal standards in 2024. Its transition timeline deprecates existing public-key algorithms by 2030 and disallows them by 2035 [27]. Major TLS implementations already negotiate post-quantum key exchange, and signature migration is expected to follow as certificate infrastructures are replaced.

Yet the final step of this transition remains hard to answer. The question an operator actually faces is which key encapsulation mechanism (KEM) and which signature algorithm to deploy between two concrete machines. Post-quantum algorithms differ on individual performance metrics by an order of magnitude. FrodoKEM’s ciphertext is roughly fourteen times larger than ML-KEM’s, and CROSS signatures are some twenty times larger than Falcon’s. Which metric dominates the handshake completion time perceived by a user depends on the hardware at both endpoints, the network between them, and TCP transport dynamics that respond discontinuously to message size.

Three properties of the problem, each confirmed by our measurements, motivate this work. Algorithm rankings do not port across hardware. In our experiments, the best signature algorithm at NIST level 3 reverses between ML-DSA-65 and MAYO-3 solely according to which machine plays the TLS server role. Transport cost grows discontinuously. A certificate chain that pushes the server’s first response flight beyond the initial TCP congestion window incurs additional delay, which we measure as a +124ms step on a link with 160ms round-trip time. And microbenchmarks cannot detect deployment obstacles. Certificates for the standardized SLH-DSA can be issued but are rejected by the TLS layer, and same-named algorithms from different provider builds fail to interoperate. A recommendation must therefore be derived from measurements taken on the machines in question and validated by real handshakes between them. Existing post-quantum TLS studies characterize the cost of fixed algorithm sets across emulated and live networks [31, 37, 38, 21]; the complementary question of which combination to deploy between a specific pair of machines, and whether it negotiates, has been outside their scope.

This paper presents a measurement-driven recommendation system built on that principle. The system probes each host’s hardware resources and crypto stack, benchmarks every candidate’s cryptographic operations on the machine that will execute them, ranks candidates with a cost model that converts both computation and transport into expected handshake contribution time in microseconds, and finally subjects the top combinations to repeated real TLS handshakes. Only candidates that pass this validation gate receive the label “recommended.” Each stage checks the others. The cost model ranks what the benchmarks measured, the gate confirms with measurements what the model ranked, and any disagreement between model and gate is reported rather than hidden. The main contributions are as follows.

- a recommendation pipeline that separates benchmark evidence from the TLS validation gate and labels every result with its provenance;

- a handshake cost model unifying computation and transport in time units, with role attribution, chain-depth scaling, and the TCP slow-start penalty;
- Pareto frontiers and significance tie bands that keep measurement noise out of the ranking;
- a cross-machine mode that composes per-endpoint measurements and adjudicates candidates over the real link; and
- an empirical study across three hardware placements, a network sweep, and chain depths, with interoperability findings that justify measurement-based validation.

We delimit the scope of this work. The system compares candidates only within a user-declared NIST security level, leaving choices between levels or security assumptions as policy decisions. Categorical security properties such as conservative design lineage or floating-point timing risk enter only as annotations and hard filters, never as score terms. Security properties and performance measurements cannot justifiably share one numeric axis. The system also evaluates algorithms as shipped by deployed crypto stacks. It is a deployment decision tool, and differences between implementations of the same scheme are part of what it measures.

The remainder of this paper is organized as follows. Section 2 provides background, and Section 3 reviews related work and positions this paper. Section 4 presents the system design, and Section 5 details the cost model and the TLS validation gate. Section 6 describes the experimental setup, and Section 7 presents the results and analysis. Section 8 concludes.

## 2 Background

### 2.1 The TLS 1.3 Handshake and Its Cost Structure

TLS 1.3 [33] completes one ephemeral key exchange and one server authentication within a single round trip. The client’s ClientHello carries a fresh KEM public key in the key\_share extension; the server’s flight carries the ciphertext, the certificate chain, and a CertificateVerify signature over the transcript made with the leaf certificate’s private key. The client decapsulates, verifies the chain up to a trusted root, and checks the transcript signature; Fig. 1 summarizes the exchange.

The cost structure follows from this flow. The KEM key share is ephemeral, so key generation, encapsulation, and decapsulation all run on every connection, and the public key and ciphertext each cross the network once. The server pays one signing operation per handshake and the client one verification per certificate in the chain, and the certificates, each carrying a public key and an issuer signature, dominate the bytes the server sends. Signature key generation happens once at certificate issuance and never during a handshake; conflating it with per-handshake cost unfairly penalizes schemes such as Falcon whose key generation is slow but whose handshake-time work is minimal. The two endpoints therefore bear different costs, and a cost model that ignores this asymmetry

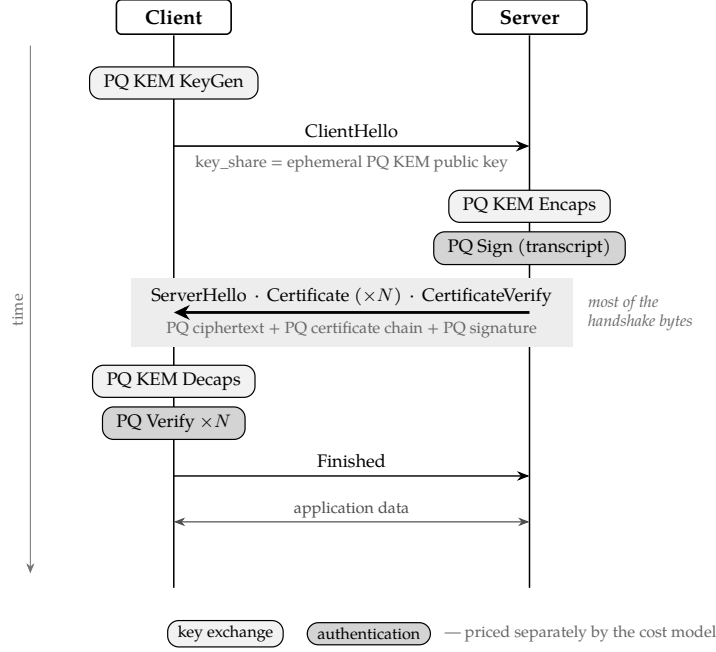


**Fig. 1.** Role-attributed cost structure of the post-quantum TLS 1.3 handshake.

implicitly assumes identical machines at both ends. Our experiments show that this assumption can flip the recommendation.

On the transport side, bytes do not convert linearly into time. During slow start a TCP sender may transmit only up to the congestion window before acknowledgments arrive, and the window doubles once per round trip [3]; the initial window is typically 10 segments, about 14.6 KB under standard Ethernet framing [11]. Post-quantum handshakes routinely exceed this threshold, and a server flight that outgrows the initial window suffers a step increase in latency equal to a full round-trip time, regardless of bandwidth. On links with hundreds of milliseconds of delay this single step exceeds every computational difference between candidates, so a ranking function must model the discontinuity, and evaluation must confirm it end to end because the effective window also depends on deployment details such as the tunnel’s maximum transmission unit (MTU).

## 2.2 Heterogeneity of the Post-Quantum Design Space

NIST standardization has centered on lattices: ML-KEM [25] standardizes CRYSTALS-Kyber [9], ML-DSA [24] standardizes CRYSTALS-Dilithium [13], the stateless hash-based SLH-DSA [26] is based on SPHINCS+ [6], and Falcon [14], with much smaller signatures from fast Fourier sampling over NTRU lattices, is slated to follow. A wider design space remains relevant to deployment. FrodoKEM [23] avoids

**Table 1.** Algorithm Candidate Pool (Sizes per the Submitted Specifications; ML-DSA-44 Is Ranked in the Level-1 Band)

| Algorithm             | Level   | pk (B)     | ct/sig (B) | Note         |
|-----------------------|---------|------------|------------|--------------|
| ML-KEM-512/768/1024   | 1/3/5   | 800–1568   | 768–1568   | FIPS 203     |
| FrodoKEM-640/976/1344 | 1/3/5   | 9616–21520 | 9720–21632 | conservative |
| BIKE-L1/L3/L5         | 1/3/5   | 1541–5122  | 1573–5154  | not selected |
| ML-DSA-44/65/87       | 2/3/5   | 1312–2592  | 2420–4627  | FIPS 204     |
| Falcon-512/1024       | 1/5     | 897–1793   | 666–1280   | FP risk      |
| MAYO-1/2/3/5          | 1/1/3/5 | 1420–5554  | 186–964    | add'l round  |
| CROSS-RSDP-128        | 1       | 77         | 13152      | add'l round  |

structured lattices at the cost of larger keys and ciphertexts, a conservative choice that Germany’s BSI includes among its recommended mechanisms [10]. BIKE [4] is a code-based KEM evaluated through round 4 but not selected, and the additional signature competition contributed designs with different size/speed trade-offs such as the multivariate MAYO [7] and the code-based CROSS [5]. This heterogeneity at the level of design families—not parameter sets—is what makes environment-specific selection nontrivial: no algorithm dominates in computation, transport bytes, and memory at once.

Candidates are compared within NIST security categories, which calibrate strength against exhaustive attacks on symmetric primitives (category 1 corresponds to a 128-bit block cipher, category 3 to 192 bits, category 5 to 256 bits, and categories 2 and 4 to hash collision resistance). Deployments fix a category first as a policy matter and select within it; our system takes the security level as input and ranks strictly within the band.

Table 1 lists the candidate pool our system covers and shows how wide the trade-off space is even among TLS-usable algorithms. The structured-lattice designs are fast and small; FrodoKEM accepts roughly a tenfold size increase for a more conservative assumption; BIKE sits between them in size but pays heavily in computation; MAYO pairs kilobyte-scale public keys with sub-kilobyte signatures; and CROSS pairs a 77-byte public key with a 13 KB signature. The asymmetry extends to operations within a scheme: Falcon verifies much faster than it signs, MAYO’s signing depends on wide SIMD units and degrades substantially without them, and ML-DSA is rarely the best at any operation but rarely far behind. An algorithm’s performance is thus a vector of operation costs, and which component matters is a property of the deployment environment.

### 2.3 Crypto Providers and Negotiation Identifiers

Post-quantum algorithms reach TLS through the crypto stack, not through protocol changes: TLS 1.3 already negotiates the key-exchange group and the signature scheme by code point, so a KEM occupies a named-group slot in `key_share` and a post-quantum signature a `signature_algorithms` entry. In OpenSSL’s provider architecture, a provider is a loadable module that registers algorithms with the library. Recent toolkit releases implement ML-KEM and ML-DSA natively; the wider candidate field arrives through the Open Quantum Safe project [39, 28],

**Table 2.** Comparison with Prior Work

| Work                      | Environment                          | Operation benchmarks | TLS handshakes | Pair-specific ranking | Validated recommendation |
|---------------------------|--------------------------------------|----------------------|----------------|-----------------------|--------------------------|
| OQS speed / pqm4 [39, 17] | fixed reference platforms            | ✓                    | —              | —                     | —                        |
| Paquin et al. [31]        | emulated networks (latency, loss)    | —                    | ✓              | —                     | —                        |
| Sikeridis et al. [37, 36] | live networks, constrained links     | —                    | ✓              | —                     | —                        |
| Sosnowski et al. [38]     | emulated networks, updated stacks    | —                    | ✓              | —                     | —                        |
| Tasopoulos et al. [40]    | Cortex-M embedded devices            | —                    | ✓              | —                     | —                        |
| Abbasi et al. [1]         | heterogeneous platforms (server-IoT) | ✓                    | ✓              | —                     | —                        |
| Montenegro et al. [21]    | emulated networks (latency, loss)    | ✓                    | ✓              | —                     | —                        |
| <b>This work</b>          | target machine pair (measured)       | ✓                    | ✓              | ✓                     | ✓                        |

whose liboqs implements a broad set of algorithms and whose oqs-provider exposes them to OpenSSL-based stacks.

Implementation alone does not make an algorithm usable in a handshake. Both endpoints must agree on its negotiation identifiers (the group and signature-scheme code points) and on the wire format of the messages they name. For classical algorithms both are fixed by IANA registries and stable specifications; for many post-quantum algorithms the code points are still experimental, chosen by the provider and changeable between releases, and a wire format can change when its specification is revised. Two endpoints therefore interoperate only when stack support, negotiation identifiers, and message formats all align.

### 3 Related Work

To the best of our knowledge, no prior work combines resource probing, on-device benchmarking, an explicit transport cost model, and an end-to-end TLS validation gate into a single recommendation pipeline; Table 2 summarizes the comparison. The migration discourse frames the coming decade as a problem of crypto-agility, but agility mechanisms answer only how to swap algorithms, not which algorithm to swap in, and that decision recurs whenever the hardware, the network, or the candidate pool changes. Existing selection guidance is largely categorical (security-level tables, size comparisons, suitability notes for embedded targets); a quantitative, environment-specific answer is not its aim. The closest approaches are the TLS benchmark studies reviewed below [31, 37, 36, 38, 21], which characterize the cost of fixed algorithm sets under controlled network conditions. This paper asks a different question: which combination should be deployed between two particular machines, and whether it actually negotiates. Answering it requires ranking candidates for a measured pair, composing measurements from both endpoints, and validating the result end to end; three of our findings surface only in that last step.

#### 3.1 Post-Quantum TLS Measurements and Protocol Variants

At the operation level, the speed utilities of Open Quantum Safe and pqm4 [39, 17] report per-operation costs on fixed reference platforms. That serves imple-

mentation comparison well; deployment decisions need more, because reference numbers do not port across instruction-set architectures and do not indicate whether the local TLS stack can negotiate an algorithm.

At the handshake level, Paquin et al. [31] evaluated post-quantum key exchange and authentication in TLS 1.3 under network emulation and showed that completion time is governed by the interaction of message size with latency and loss, not by CPU cost alone. Sikeridis et al. [37] evaluated post-quantum server authentication, confirmed that certificate size dominates on constrained links, and observed the same TCP congestion-window effect that we measure; the same authors later quantified the overhead across TLS and SSH [36]. Crockett et al. [12] systematized the design choices for integrating post-quantum and hybrid algorithms into TLS and SSH on Open Quantum Safe. Later studies extended the evaluation to updated stacks [38] and to Cortex-M-class devices [40], where the cost structure shifts again. Abbasi et al. [1] benchmarked NIST-selected algorithms from servers to IoT devices, measured TLS 1.3 handshakes on an OQS-enabled OpenSSL, and derived guidance per environment class, and Montenegro et al. [21] built an OpenSSL-based framework that measures traditional, hybrid, and post-quantum handshakes under emulated networks and reports configuration-mismatch failures as a deployment challenge. Both characterize costs for classes of environments, a different goal from ranking candidates for one measured machine pair and validating the result between those machines. These studies established handshake completion time under given network conditions as the meaningful metric; our cost model adopts the same perspective but turns it into a ranking function that can be evaluated for an arbitrary pair of machines. A separate line of work redesigns the protocol: KEMTLS [34] replaces the handshake signature with KEM-based authentication. This paper addresses selection within the standard TLS 1.3 handshake and therefore applies to deployed stacks.

### 3.2 Standardization and Industry Deployment

Hybrid groups combining classical elliptic-curve exchange with a post-quantum KEM have been standardized for TLS 1.3 [18] and are recommended by security agencies for the transition period [2]; our system prices hybrid and PQC-only modes with the same cost model. The standardized hybrid groups are now negotiated by major TLS implementations, and post-quantum key agreement has reached large messaging deployments whose protocols have since been formally analyzed [8, 20]. Each such deployment was preceded by an environment-specific measurement campaign; our system aims to automate such campaigns so that any operator can run them.

## 4 System Design

inputs to algorithm selection keep moving: implementation maturity per instruction set, the provider’s negotiation identifiers and message formats, the candidate

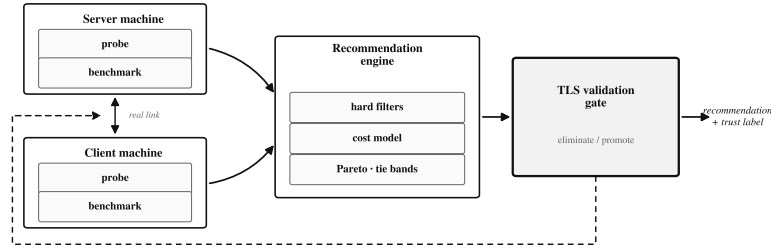


**Fig. 2.** Overview of the recommendation pipeline.

pool, the deployment topology. Three of them changed within the span of our experiments. We therefore treat selection as a procedure to be repeated rather than a decision made once, and designed the full probe–benchmark–adjudicate cycle as one automated pass over cached measurements that can be re-run whenever anything changes.

Four principles define the system. Measure rather than assume: computation costs come from benchmarks on the target machines, network parameters are measured whenever a real link exists, and public reference data serves only as an explicitly labeled fallback. Evidence and adjudication live in separate layers: operation benchmarks provide the evidence for ranking, end-to-end handshake measurements deliver the verdict on top candidates, and neither substitutes for the other. Measured and estimated values never mix within one ranking: a candidate that cannot be measured on the local stack is excluded rather than padded with reference numbers, and in cross-machine mode the pool is the intersection of what both endpoints measured. Finally, every result carries a trust label recording how it was obtained, and the model’s approximations and any skipped or truncated validation are stated in the report.

#### 4.1 Pipeline and Measurement Layers

Fig. 2 shows the four-stage pipeline. Resource probing records the facts the later stages depend on (CPU and memory, instruction-set features such as AVX2 and NEON, and the TLS toolkit and provider stack), places the machine between server-class and constrained, and identifies which algorithms the local stack can express at all. The instruction-set family is recorded because the relative cost of the same algorithm against its competitors can differ substantially between a vector-optimized x86 path and its ARM counterpart, so every ranking is tied to measurements from the family that will actually execute. Operation benchmarking measures each algorithm’s key generation, encapsulation, decapsulation, signing, and verification with the throughput benchmark built into the TLS toolkit, so the measured code path is the one a real handshake executes. The recommendation engine removes infeasible candidates with hard filters and

ranks the rest with the cost model, and the TLS validation gate reproduces real handshakes for the top candidates and delivers the verdict.

The two measurement layers play different roles. Operation benchmarks are the evidence: precise, low-noise, and able to attribute cost to individual primitives. Handshake measurements are the verdict: end-to-end, capturing everything the model might miss. Handshakes cannot serve as the primary ranking evidence, because a single handshake mixes KEM and signature costs, differences of tens of microseconds between fast algorithms drown in TCP and TLS stack overhead, and measuring every KEM $\times$ signature combination is quadratic in the pool size and infeasible on constrained devices. The system therefore ranks with operation benchmarks and reserves handshake measurements for validating a small set of top candidates.

The toolkit’s throughput benchmark runs each primitive for a fixed wall-clock interval and reports operations per second. A slower machine therefore does not take longer to benchmark, and an entire security-level band is measured in roughly one minute even on a Raspberry Pi. Standardized algorithms are measured through the toolkit’s native implementations and the remaining candidates through the quantum-safe provider. Algorithms that cannot be benchmarked locally are excluded from the measured ranking rather than approximated. Measurements are tied to the toolkit build that produced them: a toolkit change invalidates every cached benchmark, and the provider version is pinned for benchmarking, handshake trials, and certificate generation alike. The same pinning point aligns TLS code points across provider builds, a step whose necessity became apparent during the experiments.

## 4.2 Algorithm Candidate Pool

The pool follows a TLS-proven principle: only algorithms that actually complete a TLS handshake on the deployed stack qualify. Applying it to a current TLS toolkit with a quantum-safe provider yields the pool in Table 1: three KEM families and four signature families spanning structured lattices, unstructured lattices, and codes. Notably, SLH-DSA [26] is excluded by measurement, not by choice: certificates can be issued, but the TLS stack rejects them during the handshake.

Exclusions carry as much information as inclusions. Stateful hash-based signatures such as XMSS and LMS are unusable in TLS, which signs concurrently and statelessly, because they must never reuse a one-time key. Hash-function variants of included schemes are deduplicated to the most widely deployed variant. BIKE is retained, with an explicit not-selected annotation, because it is the only code-based KEM measurable on this stack and thus the comparison baseline for that design class. Schemes whose parameter sets remain in flux after published attacks, and schemes with public keys of tens of kilobytes offered at a single security level, are excluded because they offer no meaningful deployment trade-off in today’s TLS. The KEM side is narrower than the signature side because NIST standardized a single KEM and dedicated its additional round to

**Table 3.** Trust Label Taxonomy

| Label                 | Benchmarks     | TLS gate | Standing       |
|-----------------------|----------------|----------|----------------|
| Measured, validated   | target device  | run      | recommendation |
| Measured, unvalidated | target device  | skipped  | evidence only  |
| Static estimate       | reference data | —        | informational  |

signatures. Sizes, memory tiers, and risk annotations are static parameters curated from the specifications; all performance numbers are measured, and where a provider build’s measured sizes disagreed with published tables, the measured sizes were used.

### 4.3 Trust Labels

Every recommendation carries one of the three trust labels in Table 3, stating how much of the pipeline stands behind it. A result that passed the full pipeline is labeled measured, TLS-validated; one whose gate was skipped is measured, TLS-unvalidated; and one built on reference data rather than benchmarks is a static estimate, informational. A top candidate is always produced, but only the first label makes it a deployment recommendation. The label also records the age of the underlying benchmarks and, in cross-machine mode, whether both endpoints were measured.

## 5 Cost Model and TLS Validation Gate

### 5.1 Handshake Cost Model

Combining computational microseconds and transport bytes in a weighted sum requires an arbitrary exchange rate between incommensurable units, and the ranking inherits that arbitrariness. This paper instead converts bytes into time under the declared network conditions. A candidate’s expected handshake contribution is

$$T = T_{\text{compute}} + T_{\text{transport}}, \quad (1)$$

where  $T_{\text{compute}}$  is the measured computation time paid by the declared role, and the transport term is

$$T_{\text{transport}} = \frac{8B}{\beta} + (k - 1) \cdot \text{RTT}, \quad k = \left\lceil \log_2 \left( \frac{B}{W_0} + 1 \right) \right\rceil, \quad (2)$$

with  $B$  the handshake bytes the candidate adds,  $\beta$  the bandwidth,  $W_0$  the initial congestion window (10 segments, about 14.6 KB), and  $k$  the number of slow-start rounds needed to deliver  $B$ . The first term is serialization delay; the second charges one full round trip for every slow-start round beyond the first, which is the discontinuity described earlier, and vanishes for flights that fit within the

initial window.  $T$  is a differential contribution: costs common to all candidates, such as the base round trip and TCP connection establishment, are omitted, so  $T$  ranks candidates but is not an absolute handshake time. Because  $B$  counts only the candidate’s own bytes, the window test flags candidates that exceed the window by themselves; a crossing caused by the total flight of a KEM, a certificate chain, and TLS framing together is left to the validation gate.

The computation term is attributed by role. The server share is one encapsulation and one signing operation; the client share is key generation, decapsulation, and one verification per certificate in the chain. The declared role selects whose measured costs enter  $T_{\text{compute}}$ , and in cross-machine mode each share is taken from the machine that actually pays it. The declared chain depth  $N$  multiplies both the verification count and the transported certificate bytes. KEM key generation is included in the score while signature key generation is excluded and reported alongside it. Since  $T$  is additive,  $T_{\text{total}} = T_{\text{KEM}} + T_{\text{SIG}}$ , so KEM and signature rankings are computed separately, a property weighted-sum scoring cannot provide. Hybrid groups such as X25519 [19] with ML-KEM-768 [18] are priced with the same two terms: the transport term adds the curve’s byte overhead and the computation term adds a classical share measured with the toolkit’s elliptic-curve benchmark.

The model is simple enough to evaluate by hand. On a tunnel measured at 4.4 ms RTT and 24.5 Mbps, ML-KEM-768 adds 2 272 bytes, which serialize in about 742  $\mu\text{s}$  with no round-trip penalty; adding 81  $\mu\text{s}$  of role-attributed computation yields 823  $\mu\text{s}$ , about 90 percent of it transport. On fast networks, ranking among lattice KEMs is effectively decided by size.

The model makes four deliberate approximations. Following deployment practice, the level-1 band admits ML-DSA-44 although it is formally category 2. The client’s verification count is one per chain certificate, since the transcript verification common to all candidates cancels. Scores are differential contributions, not absolute times. And signature algorithms are gated on certificate usability but not compared end-to-end, a limitation revisited with the gate’s scope.

## 5.2 Deployment Profiles and Hard Filters

A profile declares a deployment scenario as physical network parameters, not as a preference-weight vector; Table 4 lists the presets, and in cross-machine mode measured values replace declared ones. Round-trip time and bandwidth are quantities an operator can state and measure, so a slow network makes the transport term dominant arithmetically and no unjustifiable weight constant enters the ranking. When no profile is declared, one is inferred from the probed hardware; in cross-machine mode the weaker endpoint governs because the bottleneck determines feasibility.

Before ranking, hard filters remove candidates that are infeasible rather than merely slow: (i) algorithms not measurable on the local stack; (ii) handshake bytes exceeding the initial congestion window, under the constrained-network and constrained-device profiles; (iii) memory tiers exceeding the platform budget, under the constrained-device profile; (iv) Falcon on unprotected embedded

**Table 4.** Deployment Profile Presets

| Profile             | RTT    | Bandwidth | Additional hard filters |
|---------------------|--------|-----------|-------------------------|
| Server              | 1 ms   | 1 Gbps    | —                       |
| Desktop             | 25 ms  | 100 Mbps  | —                       |
| Constrained device  | 50 ms  | 10 Mbps   | memory tier, FP, cwnd   |
| Constrained network | 150 ms | 1 Mbps    | congestion window       |

targets, because its floating-point signing is difficult to make constant-time there; and (v) parameter sets below the requested security level. Probing informs the filters but never adjusts scores, because a slow machine’s slowness is already present in its measured microseconds. Memory appears only in filters: the measured peak resident set of the benchmark process is dominated by the toolkit’s own overhead and does not discriminate between algorithms, so memory enters only as categorical tiers derived from published measurements such as pqm4 [17]. Keeping literature-derived data inside a binary feasibility check and out of the continuous score is a design boundary; scores contain only quantities measured on the target machine or declared by the operator.

### 5.3 Pareto Frontiers and Tie Bands

A single best candidate is a poor summary of a three-dimensional trade-off, so the ranking is accompanied by the Pareto frontier over computation, transport size, and memory; an operator who instead needs minimum bytes on a low-bandwidth link can see the reasonable alternatives with their evidence.

A point ranking can also be an artifact of noise. On fast networks rankings are decided by computation times whose run-to-run variance is a few percent, and between two similar lattice schemes that variance can exceed the gap being ranked. Two mechanisms keep noise out of the verdict. First, adjacent candidates whose score difference is below  $\max(1\ \mu\text{s}, 0.10 \times \max(T_{\text{compute}}))$  are declared a tie group; the band scales with the computation term because that is where measurement noise originates, and each candidate is compared to the group’s first member rather than its neighbor so that pairwise-close candidates cannot chain into one group. Second, the validation gate widens its scope to the entire leading tie group, so candidates separated by less than the noise floor are decided by end-to-end measurement rather than by point estimates. Tie groups occur in practice: in one level-1 placement, ML-DSA-44 (2 289  $\mu\text{s}$ ) and MAYO-1 (2 451  $\mu\text{s}$ ) fall within one band and are reported as sharing a rank.

### 5.4 TLS Validation Gate

The gate reproduces 30 handshakes per combination using the TLS toolkit’s own test server and client, after three discarded warm-up connections, and reports the median and interquartile range. A fresh client process is spawned per connection because the toolkit’s bulk timing tool cannot pin the negotiated group;

the startup overhead is a constant added to every combination and cancels in relative comparison. Three rules govern the verdict.

1. Negotiation failure: if the group cannot be negotiated or the peer alerts and aborts, the KEM is eliminated and the next-ranked KEM is promoted.
2. Certificate failure: if issuance, server-side loading, or client-side verification fails, the signature algorithm is eliminated; KEM adjudication continues with a known-good reference certificate so that one broken signature does not block KEM validation.
3. Promotion: a lower-ranked candidate overtakes a higher-ranked one only when its handshake median is faster and the interquartile ranges do not overlap.

After any elimination, the Pareto frontier and tie groups are recomputed over the survivors.

The gate does not adjudicate the whole pool, which would reintroduce the quadratic matrix the two-layer design avoids. Its scope is the top of each ranking, widened to include the entire leading tie group and extended downward by eliminations: each candidate the gate removes promotes the next, which is adjudicated in turn, so the verdict never rests on an unadjudicated candidate. By default the scope covers the top two candidates of each ranking, and lower-ranked candidates are reported as not adjudicated. One limitation must be stated: signature algorithms are gated on certificate usability only and are not compared end-to-end against each other, so the signature ranking rests on the model. End-to-end comparison of all KEM×signature combinations would make the adjudication matrix quadratic; we revisit this trade-off in the conclusion.

The gate reproduces the declared chain depth rather than assuming it. It generates a self-contained certificate-authority chain of a root, the declared number of intermediates, and a leaf, all signed with the candidate algorithm; the server presents the full chain and the client, configured with the generated root as trust anchor, treats verification failure as fatal. Verification really runs once per certificate and bytes really scale with depth: for ML-KEM-768 with ML-DSA-65, a depth-2 chain delivers 15 686 bytes to the client versus 10 183 at depth 1, a step of +5 503 bytes against the model’s +5 261, the excess being X.509 encoding overhead. The chain’s two effects—extra verification on the client CPU and extra bytes through the congestion window—are the terms the model scales with depth, and the gate measures whether the total flight crosses the window.

Loopback adjudication cannot represent a deployment between two machines: role costs collapse onto one CPU, and a near-zero round-trip time hides every transport penalty. In cross-machine mode each endpoint is probed and benchmarked separately, the computation term sums the server share measured on the server machine and the client share measured on the client machine, the candidate pool is the intersection of what both endpoints measured, and the automatic profile derives from the weaker endpoint. Network characteristics are measured over a lightweight control channel (the median of repeated round-trip probes for RTT, and the transfer time of a one-mebibyte payload for bandwidth) and replace the profile presets in Eq. (2). Adjudication then runs over the real

**Table 5.** Experimental Platforms

| Machine | CPU               | Cores       | RAM     | ISA          |
|---------|-------------------|-------------|---------|--------------|
| Mac     | Apple M2, 3.5 GHz | 8           | 8 GiB   | aarch64/NEON |
| Linux   | x86_64, 2.9 GHz   | 8 (16 thr.) | 15 GiB  | x86_64/AVX2  |
| RPi     | Pi 4B, 1.5 GHz    | 4           | 3.7 GiB | aarch64/NEON |

link with the test server and its certificates hosted on the server machine, so signature support is adjudicated against the peer’s stack and group negotiability against the intersection of both stacks, and the client verifies the remotely issued chain end to end. The recommendation is computed on the machine that will play the TLS client, matching the direction of measurement, and every stage’s output is inspectable in the report, so an unexpected recommendation can be audited back to the measurements that produced it.

## 6 Experimental Setup

### 6.1 Platforms and Scenarios

Table 5 lists the three machines, selected to span a wide performance range (about  $5\times$  for ML-KEM and far more for provider-served schemes) and two instruction-set architectures: a macOS laptop, an x86\_64 server with AVX2, and a Raspberry Pi 4 representing constrained edge hardware. All three run OpenSSL 3.6.3 [30] with the Open Quantum Safe provider [28], version-pinned as described earlier, but the provider builds are not identical: the Linux and RPi machines run oqs-provider 0.12.0-dev, whereas the macOS machine runs an older build derived from release 0.9.1. We preserve this asymmetry because a migration in progress is a set of endpoints upgraded at different times, and it is the backdrop against which the interoperability failures described later surface. The build difference affects only algorithms served by the provider on the Mac; the cross-architecture comparison that drives the level-3 reversal in Section 7.2 is between the Linux and RPi machines, which share one build.

Three hardware placements exercise cross-machine mode: **A** Linux server with Mac client, **B** Mac server with RPi client, and **C** RPi server with Mac client, covering both endpoints fast, a fast server with a constrained client, and a constrained server that assigns signing cost to the weakest CPU. The interconnect is a tunnel whose measured RTT and bandwidth vary from run to run (3.4–4.9 ms and 15–39 Mbps across the 18 runs; medians 3.6 ms and 25 Mbps), and each run’s ranking uses the values measured in that run. Network degradation is applied in scenario A with the Linux kernel’s netem emulator in two settings whose system-measured effective values are 57 ms / 8.9 Mbps and 160 ms / 1.0 Mbps, representing a degraded wide-area path and a constrained last-mile link; the system’s own measurements, not the emulator’s nominal settings, enter the model. Each scenario runs at NIST security levels 1, 3, and 5 in both PQC-only and hybrid key exchange at certificate-chain depth 1, and chain depth 2 is evaluated as a

separate level-3 experiment on every placement. In hybrid mode every KEM candidate is ranked and adjudicated as its hybrid variant: ML-KEM through the standardized X25519MLKEM768 and SecP384r1MLKEM1024 groups [18] at levels 3 and 5 and the provider’s experimental X25519–ML-KEM-512 group at level 1, and FrodoKEM and BIKE through the provider’s experimental groups.

## 6.2 Measurement Protocol

Every number in the next section comes from a complete scenario run, from probing through gate validation, executed as a single invocation of the system. Both endpoints refresh their measurement bundles before each campaign; within a campaign, cached benchmarks under one day old are reused, and the network is re-measured on every invocation. The gate performs 30 handshakes per adjudicated combination after discarding warm-up connections and reports the median, interquartile range, and failure count; combinations with systematic failures are handled by the elimination rules rather than averaged. The promotion rule’s requirement of non-overlapping interquartile ranges is conservative at this sample size by design. Measured medians include a constant per-connection overhead from spawning a fresh test client, identical across combinations, so medians are meaningful relative to one another within a scenario, and predicted contributions, which are differential by construction, are compared with measured differences rather than with medians.

## 7 Results and Analysis

Table 6 presents the complete evaluation matrix under the desktop profile at chain depth 1: the recommended KEM–signature combination and the measured handshake medians in both key-exchange modes for each placement and security level. In all 18 scenario runs the recommended combination passes gate validation with zero failures out of 30 handshakes.

### 7.1 Per-Placement Results

**Scenario A: Linux Server and Mac Client** With both endpoints fast, medians sit at 49–54 ms across all levels and modes, computation terms are in the hundreds of microseconds, and ranking is decided largely by transport bytes. ML-KEM-768 is recommended at level 3 (predicted 0.82 ms, measured 52.0 ms); FrodoKEM-976 ranks second in the model but fails every handshake because of the wire-format incompatibility described later, and its elimination promotes BIKE-L3 at 95.6 ms. MAYO-3 is recommended for the level-3 signature because its smaller certificate (3.7 KB versus 5.3 KB for ML-DSA-65) carries it to 1.65 ms versus 2.74 ms. Falcon ranks first at levels 1 and 5, while CROSS remains transport-bound even on this fast link: 3.5 ms of its 4.62 ms total is transport for its 13 KB signature. Hybrid key exchange stays within measurement noise at every level; at level 3 the gate confirms the model’s prediction with medians of 53.4 versus 52.0 ms.

**Table 6.** Recommended Combination and Measured Handshake Median (ms) per Placement and Level

| Sc. | Lvl | Recommended combination   | PQC-only | Hybrid             |
|-----|-----|---------------------------|----------|--------------------|
| A   | 1   | ML-KEM-512 + Falcon-512   | 49.2     | 50.9               |
|     | 3   | ML-KEM-768 + MAYO-3       | 52.0     | 53.4               |
|     | 5   | ML-KEM-1024 + Falcon-1024 | 49.6     | 54.0               |
| B   | 1   | ML-KEM-512 + Falcon-512   | 147.8    | 153.2              |
|     | 3   | ML-KEM-768 + ML-DSA-65    | 148.4    | 147.4              |
|     | 5   | ML-KEM-1024 + Falcon-1024 | 146.1    | 155.1 <sup>†</sup> |
| C   | 1   | ML-KEM-512 + Falcon-512   | 53.3     | 58.0               |
|     | 3   | ML-KEM-768 + ML-DSA-65    | 55.9     | 51.8               |
|     | 5   | ML-KEM-1024 + Falcon-1024 | 57.4     | 69.1 <sup>†</sup>  |

<sup>†</sup>interquartile ranges of the two modes do not overlap.

**Scenario B: Mac Server and Raspberry Pi Client** Medians rise to 146–155 ms, set not by algorithm choice but by the RPi client’s per-connection processing, and the ranking changes with the division of labor. The level-3 signature recommendation is ML-DSA-65 (2.91 ms versus 5.71 ms for MAYO-3, whose verification cost is now paid on the slowest CPU). At level 1 Falcon-512 remains first and the tie-group example of Section 5.3 appears: ML-DSA-44 and MAYO-1 differ by 162  $\mu$ s, inside the 174  $\mu$ s band, and share a rank. ML-KEM-768 is again the KEM; FrodoKEM-976 fails every attempt, and BIKE-L3 records 498.2 ms, +350 ms over ML-KEM-768. Hybrid mode is indistinguishable from PQC-only at levels 1 and 3 and separates by +9.0 ms at level 5.

**Scenario C: Raspberry Pi Server and Mac Client** Signing falls to the weakest CPU while the client side stays fast; medians span 52–69 ms. The level-3 signature recommendation remains ML-DSA-65 (6.07 ms versus 7.38 ms), Falcon-1024’s level-5 margin widens to 2.8 $\times$  over ML-DSA-87, and MAYO-5’s computation term reaches 14.0 ms. This placement produces the campaign’s largest model deviation, BIKE-L3’s measured overhead exceeding the prediction by 5.7 $\times$ , which the gate catches and the next subsection analyzes, and the campaign’s largest hybrid overhead, +11.7 ms at level 5.

The placement also exercises the profile filters. Fig. 3 contrasts the level-1 pool under the desktop profile with the automatic profile, which resolves to the constrained-device preset because the weaker endpoint is the RPi. The constrained-device filters remove BIKE-L1 and CROSS on memory tier, FrodoKEM-640 on memory and on its 19.3 KB handshake against the 14.6 KB window, and Falcon-512 on floating-point risk. Under the desktop profile every candidate passes the filters, but the gate then eliminates BIKE-L1 and FrodoKEM-640, whose handshakes all fail. The filters encode declared platform constraints and the gate discovers measured incompatibilities; neither subsumes the other.

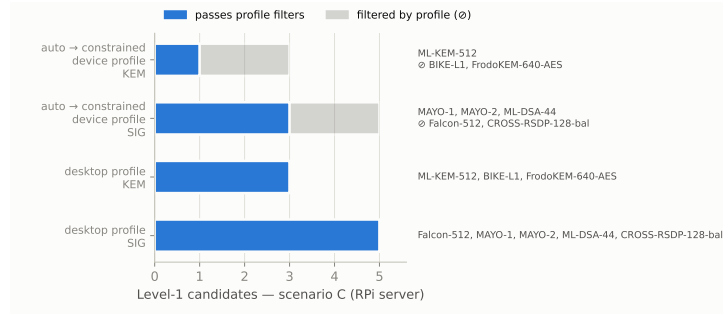


**Fig. 3.** Reshaping of the level-1 candidate pool by profile hard filters (scenario C; ø marks removed algorithms).

**Table 7.** Recommended Signature per Scenario and Level

|   | Lvl A: Linux srv | B: Mac srv       | C: RPi srv       |
|---|------------------|------------------|------------------|
|   | × Mac cli        | × RPi cli        | × Mac cli        |
| 1 | Falcon-512       | Falcon-512       | Falcon-512       |
| 3 | <b>MAYO-3</b>    | <b>ML-DSA-65</b> | <b>ML-DSA-65</b> |
| 5 | Falcon-1024      | Falcon-1024      | Falcon-1024      |

## 7.2 Cross-Placement Analysis

As Table 7 summarizes, the recommended KEM is ML-KEM everywhere and the level-1 and level-5 signatures are consistently Falcon, but the level-3 signature reverses with the placement. This subsection analyzes that reversal, the KEM ranking and its model deviation, and the placement dependence of the hybrid decision.

Fig. 4 decomposes the level-3 signature decision by placement. When the fast AVX2 server signs, MAYO-3’s smaller certificate decides the outcome; once the RPi is part of the pair, MAYO-3’s computation term rises to 4.6 ms and 6.1 ms in scenarios B and C, against 1.4 ms and 4.2 ms for ML-DSA-65, and ML-DSA-65 becomes the recommendation. The reversal is entirely a property of the pair: the same two algorithms, tunnel, and security level change order solely because the division of labor between CPUs changes, an outcome no hardware-agnostic static table can express. The reversal is a ranking-level result: the predicted gaps of 1.1 to 2.8 ms are smaller than the interquartile range of the recommended combination’s handshake medians (47.7–55.0 ms in A at level 3), so the gate confirms that both signatures negotiate but does not resolve the difference end to end.

Fig. 5 shows the operation costs behind the reversal. MAYO-3 signs  $2.8\times$  faster than ML-DSA-65 on the x86 server (322 versus 913  $\mu$ s), where wide SIMD units benefit its vectorizable arithmetic, but on the RPi it signs  $1.5\times$  slower (5 964 versus 4 107  $\mu$ s) and verifies  $5.5\times$  slower (4 329 versus 794  $\mu$ s). Neither

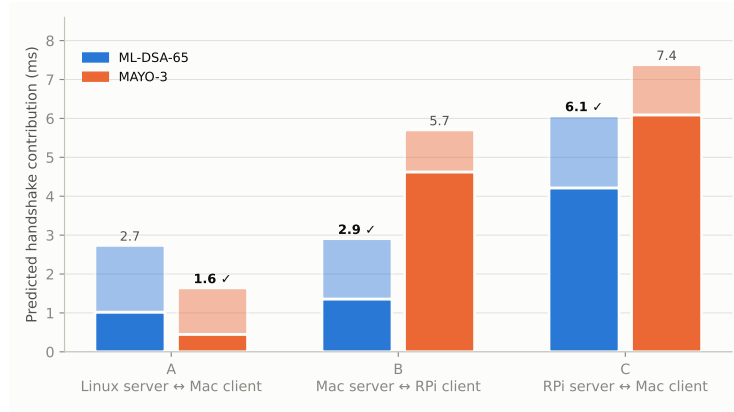


**Fig. 4.** Level-3 signature score decomposition by hardware placement (solid: computation term; translucent: transport term; check marks: recommended).

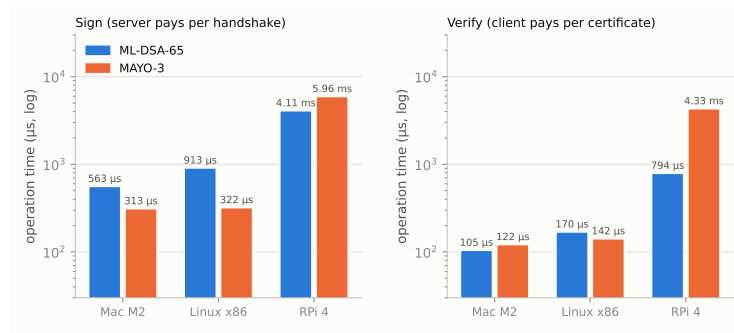


**Fig. 5.** Measured level-3 signature operation costs on the three platforms (log scale).

algorithm is uniformly superior at any operation across the three machines; the ranking can only be settled by measuring the machine that will play each role.

Table 8 collects the signature scores for all three levels. At level 1 the five schemes respond differently to placement: CROSS’s cost is dominated by signature bytes and tracks the network rather than the hardware, the MAYO variants trade public-key size against signature size, and ML-DSA-44 never ranks first, trailing Falcon-512 by 1.0 to 2.7 ms. At level 5 the gaps widen as the hardware weakens: Falcon-1024’s margin over ML-DSA-87 grows from  $2.6\times$  in A to  $2.8\text{--}2.9\times$  in the RPi-involved placements, and MAYO-5’s computation term grows to roughly nine times Falcon’s in C. Two caveats apply to Falcon’s lead. It carries the floating-point risk annotation and is removed by the hard filter under the constrained-device profile, and its key generation, excluded from the handshake score as a one-time issuance cost, is reported alongside so that an operator issuing short-lived certificates at high frequency can see it.

**Table 8.** Predicted Signature Contribution (ms) per Scenario and Level

| Lvl | Algorithm   | A           | B           | C           |
|-----|-------------|-------------|-------------|-------------|
| 1   | Falcon-512  | <b>0.63</b> | <b>0.85</b> | <b>1.32</b> |
|     | MAYO-1      | 0.70        | 2.45        | 2.90        |
|     | MAYO-2      | 1.45        | 2.99        | 3.30        |
|     | ML-DSA-44   | 1.63        | 2.29        | 4.04        |
|     | CROSS       | 4.62        | 8.12        | 8.82        |
| 3   | MAYO-3      | <b>1.65</b> | 5.71        | 7.38        |
|     | ML-DSA-65   | 2.74        | <b>2.91</b> | <b>6.07</b> |
| 5   | Falcon-1024 | <b>1.14</b> | <b>1.36</b> | <b>3.23</b> |
|     | ML-DSA-87   | 2.93        | 3.88        | 9.03        |
|     | MAYO-5      | 2.52        | 12.09       | 17.57       |

**Table 9.** Measured Level-3 KEM Operation Times per Machine ( $\mu$ s, key-gen / encaps / decaps)

| Algorithm    | Mac M2               | Linux x86       | RPi 4                  |
|--------------|----------------------|-----------------|------------------------|
| ML-KEM-768   | 31 / 20 / 31         | 32 / 19 / 29    | 145 / 82 / 125         |
| FrodoKEM-976 | 443 / 558 / 523      | 538 / 746 / 703 | 47.2k / 47.9k / 47.6k  |
| BIKE-L3      | 19.3k / 1.0k / 14.9k | 617 / 90 / 1.8k | 161.5k / 8.2k / 134.7k |

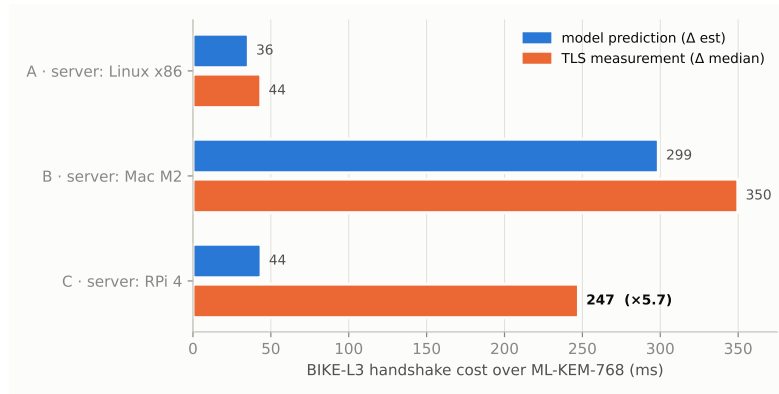
On the KEM side, Table 10 consolidates all three levels. ML-KEM is recommended at every placement and level with a predicted contribution of at most 1.9ms, and the alternatives trail by factors of several to several hundred. FrodoKEM fails every adjudicated handshake, so the baseline for the non-lattice families is BIKE, whose penalty grows with both level and hardware, from +44ms for BIKE-L3 in scenario A to +958ms for BIKE-L5 in scenario B. The level-3 operation measurements in Table 9 contain a further finding: not even the direction of a performance difference ports across instruction-set families. BIKE-L3 key generation takes  $617\mu$ s on the x86 server but  $19\,307\mu$ s on the macOS laptop,  $31\times$  slower on the machine that is faster at nearly every other primitive, because the scheme’s optimized code path targets AVX2 and has no NEON counterpart of equal maturity. A ranking extrapolated from x86 reference numbers to an ARM deployment would misjudge BIKE’s computation term by more than an order of magnitude.

Fig. 6 quantifies model accuracy for the BIKE-L3 penalty over ML-KEM-768. In scenarios A and B the measured/predicted ratios are 1.22 and 1.17; in scenario C, with the RPi as server, the measured overhead reaches  $5.7\times$  the prediction (247.5ms versus 43.7ms). The pattern holds across all 14 pairs the gate adjudicated at levels 3 and 5, in both key-exchange modes, and on the emulated links: the ratio stays between 0.98 and 1.41 except in the four RPi-server runs, where it is 5.3 to 5.9. This is why the gate exists: the model is least accurate where the weakest machine carries the heaviest role, the gate corrects

**Table 10.** KEM Field per Level and Scenario: Predicted Contribution and Measured Handshake Median (ms)

| Lvl | KEM           | A           |             | B           |              | C           |             |
|-----|---------------|-------------|-------------|-------------|--------------|-------------|-------------|
|     |               | pred.       | meas.       | pred.       | meas.        | pred.       | meas.       |
| 1   | ML-KEM-512    | <b>0.47</b> | <b>49.2</b> | <b>0.80</b> | <b>147.8</b> | <b>0.64</b> | <b>53.3</b> |
|     | FrodoKEM-640  | 10.1        | fail        | 52.3        | fail         | 31.6        | fail        |
|     | BIKE-L1       | 11.8        | fail        | 96.1        | fail         | 14.7        | fail        |
| 3   | ML-KEM-768    | <b>0.82</b> | <b>52.0</b> | <b>0.96</b> | <b>148.4</b> | <b>0.94</b> | <b>55.9</b> |
|     | FrodoKEM-976  | 16.4        | fail        | 107.8       | fail         | 63.4        | n.a.        |
|     | BIKE-L3       | 36.4        | 95.6        | 299.7       | 498.2        | 44.7        | 303.4       |
| 5   | ML-KEM-1024   | <b>0.84</b> | <b>49.6</b> | <b>1.28</b> | <b>146.1</b> | <b>1.90</b> | <b>57.4</b> |
|     | FrodoKEM-1344 | 16.7        | fail        | 196.0       | fail         | 118.6       | n.a.        |
|     | BIKE-L5       | 89.0        | 143.3       | 746.1       | 1103.6       | 112.7       | 711.2       |

fail: every handshake failed; n.a.: ranked third, outside the gate’s scope.

**Fig. 6.** Predicted versus measured BIKE-L3 overhead relative to ML-KEM-768. On the slowest server (scenario C) the measurement deviates from the model by a factor of 5.7.

it with measurement at that point, and the final recommendation rests on the gate’s measurements.

The hybrid comparison, too, resolves only across placements. At levels 1 and 3 the interquartile ranges of the two modes overlap in every placement, so by the gate’s own criterion the difference does not rise above noise, and the system recommends hybrid mode where policy permits, in line with security-agency guidance [2]. Level 5 is the exception (Table 6): the standard level-5 hybrid group pairs ML-KEM-1024 with the P-384 curve, whose measured scalar-multiplication share is far more expensive than X25519’s, and in the RPi-involved placements the overhead grows to 9.0ms in B and 11.7ms in C with disjoint interquartile ranges. Had the classical share been assumed rather than measured, the near-parity at levels 1 and 3 would have been extrapolated to level 5 and scenario C’s 20-percent overhead would have gone unpredicted.

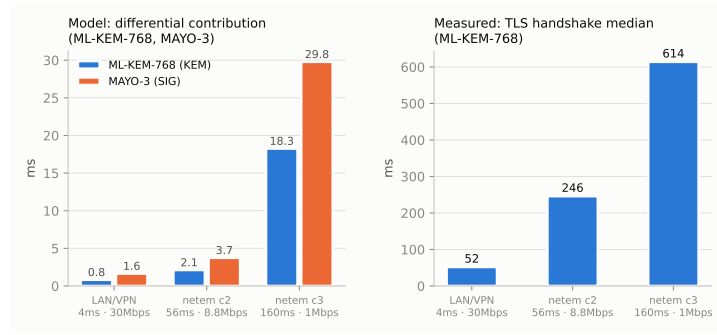


**Fig. 7.** Declared-network sweep at level 3 (left: model, right: measured handshakes). As the link degrades, the transport term comes to dominate the ranking.

### 7.3 Transport Effects

Fig. 7 sweeps the network at level 3 from the tunnel baseline to the two emulated settings. As the link degrades to 160 ms / 1 Mbps, ML-KEM-768’s predicted differential contribution grows from 0.8 ms to 18.3 ms and its measured median from 52 ms to 614 ms: the transport term moves from negligible to dominant, which is the regime change the profile mechanism is designed to declare. On the worst link the byte-heavy candidates separate sharply, BIKE-L3 completing at 706 ms (+92 ms over ML-KEM-768) and FrodoKEM-976’s predicted contribution reaching 413 ms before the gate is consulted. The ranking itself is a stable function of the declared network: ML-KEM-768 and MAYO-3 remain the level-3 recommendation at every setting, and only the margins move. At level 1 under the intermediate setting, every handshake attempt with BIKE-L1 and FrodoKEM-

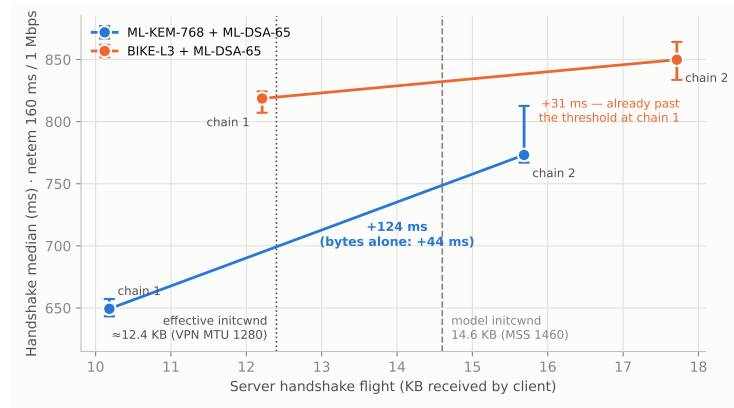


**Fig. 8.** Crossing the initial congestion window incurs a step increase in handshake latency (160 ms / 1 Mbps link, signature fixed to ML-DSA-65, chain depths 1 and 2).

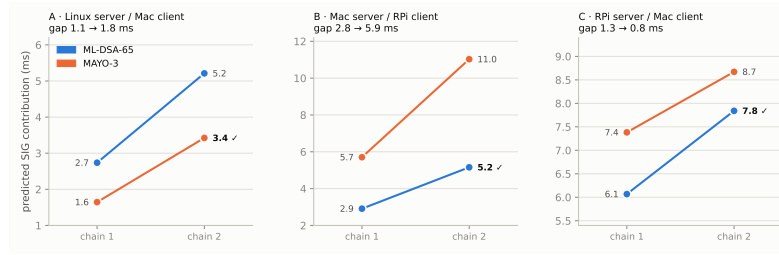


**Fig. 9.** Level-3 signature scores at chain depths 1 and 2 (panel subtitles: gap between the two algorithms; check marks: recommendation at depth 2).

640 fails, and the gate leaves ML-KEM-512 with Falcon-512 as the validated recommendation.

Fig. 8 isolates the slow-start term of Eq. (2) on the 160 ms / 1 Mbps link by varying chain depth with the signature fixed to ML-DSA-65. Growing the ML-KEM-768 flight from 10.2 KB at depth 1 to 15.7 KB at depth 2 crosses the initial congestion window and costs +124 ms—far more than the +44 ms that serialization alone would explain. The residual of about 80 ms is the slow-start stall waiting for the first window’s acknowledgments; it is smaller than the 160 ms round trip because the stall overlaps with the serialization of the first window itself. BIKE-L3, already past the threshold at depth 1, pays only +31 ms for the same growth. The measurements also reveal an effective window of about 12.4 KB, smaller than the modeled 14.6 KB because the tunnel’s MTU shrinks the segment size, a deployment detail a model fed nominal parameters would miss.

Doubling the chain depth doubles both the verification count and the certificate bytes, and Fig. 9 shows that its effect is asymmetric across placements. In scenario A, depth 2 widens MAYO-3’s advantage over ML-DSA-65 from 1.65 versus 2.74 ms to 3.42 versus 5.21 ms because the certificate-size advantage compounds; in scenario B the opposite occurs, as MAYO-3’s verification now runs twice on the RPi client and its score roughly doubles to 11.03 ms while ML-DSA-65 grows only to 5.16 ms. The recommendation set of Table 7 is unchanged at depth 2, but the gaps shift by up to a factor of two, which matters for operators near a profile boundary. The model captures slow start but neither packet loss nor TLS record-layer framing; on lossy links, retransmission dynamics can enlarge the model error, and absorbing that residual is the gate’s role, as scenario C’s BIKE deviation shows. Explicit modeling of loss dynamics is left as future work.

#### 7.4 Interoperability Findings

Three obstacles caught by the gate are the main evidence for measurement-based validation. Each has a documented upstream cause; what the gate adds is detecting it automatically for a given machine pair. In all three cases the

operation benchmarks complete normally, so no benchmark-only methodology could observe these failures.

**Standardized but Undeployable** SLH-DSA [26] certificates can be issued with the current toolkit, yet the TLS stack rejects them during the handshake as an unknown certificate type. This is a gap in protocol integration rather than a defect of our configuration. SignatureScheme code points for SLH-DSA exist in the IANA registry [16], but they were registered from an individual Internet-Draft [32] that expired without working-group adoption and carry the not-recommended flag, and the toolkit’s TLS layer does not implement them; ML-DSA, by contrast, has a working-group specification [15] and is enabled in the same toolkit’s TLS defaults. SLH-DSA’s operation costs are perfectly measurable, so a pipeline that ranked by them alone would recommend an undeployable algorithm. The gate attributes the failure to the signature and excludes it. FIPS standardization, in short, does not imply TLS deployability—a distinction only an end-to-end check can draw.

**Code-Point Drift across Provider Builds** Between two provider builds, the same group name maps to different TLS negotiation code points, producing handshake alert failures between endpoints that nominally both support the algorithm. The renumbering is a documented upstream change: the provider’s release notes state that the release in question changed FrodoKEM’s code points [29]. We extend the version-pinning mechanism to align both endpoints’ code points explicitly as part of the pinned configuration.

**Wire-Format Generations** Even with code points aligned, FrodoKEM handshakes between different library generations fail during decapsulation because one side’s ciphertext is 48 bytes larger, 15 792 versus 15 744. The difference is exactly the 384-bit salt that a revision of the FrodoKEM specification added to encapsulation to address multi-ciphertext attacks [42, 41]. No configuration alignment can resolve a message-format divergence; the gate correctly removes the combination for this machine pair, and loopback experiments, where both sides share one implementation, remain the only setting in which the scheme can be measured.

Together these findings support the design thesis: an algorithm name is not an interoperability guarantee. Stack support, agreement on negotiation identifiers, and agreement on message formats are three independent layers, and a failure in any one of them is invisible to benchmarks yet fatal to deployment. The build mismatch is not itself the predictor: across the same two builds, ML-KEM, ML-DSA, Falcon, MAYO, BIKE-L3, and BIKE-L5 all negotiate, whereas FrodoKEM, whose code points and wire format have both changed, does not. BIKE-L1 also fails every handshake in every placement; we have not isolated its cause, and the gate excludes it on the failure alone. Recommendations must therefore be validated by the actual pair of stacks that will execute them. None of these

failures is discoverable by benchmarking, and none is hypothetical: all three occur on three commodity machines within a single experimental campaign.

## 8 Conclusion

Selecting a post-quantum algorithm for a real deployment is not a table lookup; it is a measurement problem coupled with a validation problem. This paper presented a recommendation system that treats it as both: environment measurement, physically grounded cost modeling, and end-to-end TLS validation operate as distinct and mutually checking stages, and every output carries a trust label stating how much of the pipeline stands behind it.

The empirical results are consistent with the design. Across three hardware placements the system produces different recommendations from the same candidate pool, reversing the level-3 signature choice with the server hardware, and consistent recommendations where the evidence is consistent, selecting ML-KEM at every placement, level, and network. The validation gate exposes deployment obstacles that benchmarks do not reveal: a standardized signature scheme whose certificates are issued but rejected by the TLS stack, negotiation identifiers drifting between provider builds, and a wire-format divergence between generations of the same crypto provider. The measured cost of crossing the initial congestion window, +124ms on a 160ms link, shows why TCP dynamics belong in the ranking function and why the gate must measure the total flight that the per-candidate model cannot see.

Four operational observations follow. ML-KEM is the recommended KEM in every environment measured, with the alternatives trailing by factors rather than percentages. The signature choice depends on the server hardware at level 3 and on platform protection at levels 1 and 5, so a signature decision requires knowing where signing and verification will run. Hybrid key exchange adds nothing distinguishable from noise at levels 1 and 3 and is a reasonable default where policy permits, but at level 5 the standard hybrid group’s P-384 share raises the overhead to 20 percent on constrained hardware. And certificate-chain depth is a cost parameter in its own right; near the initial congestion window it changes handshake time by more than the choice between the top two signature schemes. The procedure that produces these observations is inexpensive: a security-level band benchmarks in about a minute even on a Raspberry Pi, and a full scenario run including gate validation finishes in minutes, so re-running the selection after a toolkit upgrade, certificate rotation, or topology change is a routine step.

The concrete rankings reported here are properties of the measured machine pairs and do not transfer to other hardware; this is the premise of the work rather than a limitation of it. What we expect to generalize are the structural conclusions: the reversal mechanism by which computation-dominated and transport-dominated regimes trade places, the congestion-window discontinuity, the division of labor between model and gate, and the finding that stack support, negotiation identifiers, and wire formats are three independent layers at which same-named algorithms can fail to interoperate. What transfers is the procedure:

any pair of machines that can run a TLS toolkit can be probed, measured, and adjudicated with one invocation. Two limitations remain. The signature ranking rests on the model, since the gate checks signatures only for feasibility, and end-to-end comparison among signatures would require a quadratic adjudication matrix; and this campaign spans three machines, two instruction sets, and one TLS toolkit lineage, so other toolkits may exhibit other integration defects.

While implementations, stacks, and wire formats continue to change, an algorithm recommendation is a property of a specific environment with a limited shelf life. A system that measures the environment, states its assumptions, and validates its conclusions end to end is, we argue, a sound basis for deployment guidance during the migration period.

## References

1. Abbasi, M., Cardoso, F., Váz, P., Silva, J., Martins, P.: A practical performance benchmark of post-quantum cryptography across heterogeneous computing environments. *Cryptography* **9**(2), 32 (2025)
2. Agence nationale de la sécurité des systèmes d'information: ANSSI views on the post-quantum cryptography transition (2023 follow up). Position paper, <https://messervices.cyber.gouv.fr/guides/en-follow-position-paper-post-quantum-cryptography> (2023)
3. Allman, M., Paxson, V., Blanton, E.: TCP congestion control. IETF RFC 5681 (2009)
4. Aragon, N., Barreto, P.S.L.M., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.C., Gaborit, P., Ghosh, S., Gueron, S., Güneysu, T., et al.: BIKE: Bit flipping key encapsulation. Submission to the NIST Post-Quantum Cryptography Standardization Process, Round 4, <https://bikesuite.org/> (2022)
5. Baldi, M., Battaglioni, M., Chiaraluce, F., Horlemann, A.L., Persichetti, E., Santini, P., Weger, V., et al.: CROSS: Codes and restricted objects signature scheme. Submission to the NIST PQC Standardization of Additional Digital Signature Schemes, <https://csrc.nist.gov/projects/pqc-dig-sig> (2023)
6. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS+ signature framework. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS). ACM (2019)
7. Beullens, W.: MAYO: Practical post-quantum signatures from oil-and-vinegar maps. In: Selected Areas in Cryptography (SAC 2021). LNCS, vol. 13203, pp. 355–376. Springer (2022)
8. Bhargavan, K., Jacomme, C., Kiefer, F., Schmidt, R.: Formal verification of the PQXDH post-quantum key agreement protocol for end-to-end secure messaging. In: Proceedings of the 33rd USENIX Security Symposium. pp. 469–486. USENIX Association (2024)
9. Bos, J., Lucas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS – Kyber: A CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE (2018)
10. Bundesamt für Sicherheit in der Informationstechnik: Cryptographic mechanisms: Recommendations and key lengths. BSI Technical Guideline TR-02102-1, Version 2026-01, <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf> (2026)

11. Chu, J., Dukkupati, N., Cheng, Y., Mathis, M.: Increasing TCP's initial window. IETF RFC 6928 (2013)
12. Crockett, E., Paquin, C., Stebila, D.: Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. Cryptology ePrint Archive, Paper 2019/858 (2019)
13. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(1) (2018)
14. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon: Fast-Fourier lattice-based compact signatures over NTRU. Submission to the NIST Post-Quantum Cryptography Standardization Process, <https://falcon-sign.info/> (2020)
15. Hollebeek, T., Schmieg, S., Westerbaan, B.: Use of ML-DSA in TLS 1.3. IETF Internet-Draft, draft-ietf-tls-mldsa-05, work in progress (2026)
16. Internet Assigned Numbers Authority: Transport layer security (TLS) parameters: SignatureScheme registry. <https://www.iana.org/assignments/tls-parameters/>, accessed: 2026-09-08
17. Kamnitscher, M.J., Rijneveld, J., Schwabe, P., Stoffelen, K.: pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4. Cryptology ePrint Archive, Paper 2019/844 (2019)
18. Kwiatkowski, K., Kampanakis, P., Westerbaan, B., Stebila, D.: Post-quantum traditional (PQ/T) hybrid key agreement mechanisms for TLS 1.3. IETF RFC 10024 (2026)
19. Langley, A., Hamburg, M., Turner, S.: Elliptic curves for security. IETF RFC 7748 (2016)
20. Linker, F., Sasse, R., Basin, D.: A formal analysis of Apple's iMessage PQ3 protocol. In: Proceedings of the 34th USENIX Security Symposium. pp. 5015–5034. USENIX Association (2025)
21. Montenegro, J.A., Rios, R., Lopez-Cerezo, J.: A performance evaluation framework for post-quantum TLS. Future Generation Computer Systems **175**, 108062 (2026)
22. Mosca, M.: Cybersecurity in an era with quantum computers: Will we be ready? IEEE Security & Privacy **16**(5), 38–41 (2018)
23. Naehrig, M., Alkim, E., Bos, J.W., Ducas, L., Easterbrook, K., LaMacchia, B., Longa, P., Mironov, I., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: FrodoKEM: Learning with errors key encapsulation. Submission to the NIST Post-Quantum Cryptography Standardization Process, <https://frodokem.org/> (2021)
24. National Institute of Standards and Technology: Module-lattice-based digital signature standard. FIPS PUB 204, <https://doi.org/10.6028/NIST.FIPS.204> (2024)
25. National Institute of Standards and Technology: Module-lattice-based key-encapsulation mechanism standard. FIPS PUB 203, <https://doi.org/10.6028/NIST.FIPS.203> (2024)
26. National Institute of Standards and Technology: Stateless hash-based digital signature standard. FIPS PUB 205, <https://doi.org/10.6028/NIST.FIPS.205> (2024)
27. National Institute of Standards and Technology: Transition to post-quantum cryptography standards. NIST IR 8547 (Initial Public Draft), <https://doi.org/10.6028/NIST.IR.8547.ipd> (2024)
28. Open Quantum Safe Project: oqs-provider: OpenSSL 3 provider for quantum-safe cryptography. <https://github.com/open-quantum-safe/oqs-provider>, accessed: 2026-09-08

29. Open Quantum Safe Project: oqs-provider release 0.8.0 release notes. <https://github.com/open-quantum-safe/oqs-provider/releases/tag/0.8.0> (2024), updates IANA code points for ML-KEM and changes FrodoKEM code points. Accessed: 2026-09-08
30. OpenSSL Project: OpenSSL: Cryptography and SSL/TLS toolkit. <https://www.openssl.org/>, accessed: 2026-09-08
31. Paquin, C., Stebila, D., Tamvada, G.: Benchmarking post-quantum cryptography in TLS. In: Post-Quantum Cryptography (PQCrypto 2020). LNCS, vol. 12100, pp. 72–91. Springer (2020)
32. Reddy, K. T., Hollebeek, T., Gray, J., Fluhrer, S.: Use of SLH-DSA in TLS 1.3. IETF Internet-Draft, draft-reddy-tls-slhdsa-02 (expired, not adopted) (2025)
33. Rescorla, E.: The transport layer security (TLS) protocol version 1.3. IETF RFC 8446 (2018)
34. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS). ACM (2020)
35. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* **26**(5), 1484–1509 (1997)
36. Sikeridis, D., Kampanakis, P., Devetsikiotis, M.: Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH. In: Proceedings of the 16th International Conference on Emerging Networking Experiments and Technologies (CoNEXT). ACM (2020)
37. Sikeridis, D., Kampanakis, P., Devetsikiotis, M.: Post-quantum authentication in TLS 1.3: A performance study. In: Network and Distributed System Security Symposium (NDSS) (2020)
38. Sosnowski, M., Wiedner, F., Hauser, E., Steger, L., Schoinianakis, D., Gallenmüller, S., Carle, G.: The performance of post-quantum TLS 1.3. In: Companion of the 19th International Conference on Emerging Networking Experiments and Technologies (CoNEXT Companion). ACM, Paris, France (2023)
39. Stebila, D., Mosca, M.: Post-quantum key exchange for the internet and the open quantum safe project. In: Selected Areas in Cryptography (SAC 2016). LNCS, vol. 10532, pp. 14–37. Springer (2017)
40. Tasopoulos, G., Li, J., Fournaris, A.P., Zhao, R.K., Sakzad, A., Steinfeld, R.: Performance evaluation of post-quantum TLS 1.3 on resource-constrained embedded systems. In: Information Security Practice and Experience (ISPEC 2022). LNCS, vol. 13620, pp. 432–451. Springer (2022)
41. The FrodoKEM Team: Annex on FrodoKEM updates. <https://frodokem.org/files/FrodoKEM-annex-20230418.pdf> (2023)
42. The FrodoKEM Team: FrodoKEM: Learning with errors key encapsulation — preliminary standardization proposal. <https://frodokem.org/files/FrodoKEM-standard-proposal-20230314.pdf> (2023)