Signature Placement in Post-Quantum TLS Certificate Hierarchies: An Experimental Study of ML-DSA and SLH-DSA in TLS 1.3 Authentication

José Luis Delgado
Universitat Oberta de Catalunya
jdelgado13@uoc.edu

Abstract

Post-quantum migration in TLS 1.3 couples signature-algorithm choice with certificate-hierarchy structure, chain exposure during the handshake, and role-dependent cryptographic cost. In certificate-based authentication, the practical effect of a signature family depends on where it appears in the certification hierarchy, how much of that hierarchy is exposed during the handshake, and how the resulting cryptographic cost is distributed across client and server roles. Post-quantum TLS migration must therefore be evaluated as cryptographic design within authenticated key establishment, with algorithm selection assessed in its deployment context.

This paper presents a local experimental study of TLS 1.3 authentication strategies implemented with OpenSSL 3 and oqsprovider. Using a reproducible laboratory setting, it compares ML-DSA and SLH-DSA across multiple certificate placements, hierarchy depths, and key-exchange modes, including classical, hybrid, and pure post-quantum configurations. The analysis is organized into four complementary campaigns: a leaf-only comparison, a full hierarchy strategy matrix, a depth comparison, and a key-exchange exploration.

Across the experimental matrix, the main discontinuity appears when SLH-DSA is placed in the server leaf certificate. In that configuration, handshake latency and server-side compute cost increase by orders of magnitude, whereas strategies that confine SLH-DSA to upper trust layers and preserve ML-DSA in the interactive leaf remain within a more plausible operational range. The results also show that transport size alone does not explain the heavy regime: outside leaf-SLH scenarios, transferred bytes and observed chain size track latency closely, but once SLH-DSA reaches the leaf, server-side cryptographic cost becomes dominant.

The paper evaluates post-quantum TLS migration as a problem of certificate-hierarchy design, chain exposure, and cryptographic cost concentration during live authentication. In practical terms, signature placement matters at least as much as signature-family choice.

Keywords: TLS 1.3, post-quantum cryptography, post-quantum authentication, ML-DSA, SLH-DSA, ML-KEM, certificate hierarchies, X.509, PKI, authenticated key establishment.

Contents

1 Introduction **5**

1.1 Post-quantum TLS migration as protocol and PKI design 5

1.2 Why signature placement matters 5

1.3 Research gap 5

1.4 Research questions and thesis 6

1.5 Contributions 6

1.6 Paper organization 7

2 Cryptographic and Protocol Background **7**

2.1 TLS 1.3 certificate authentication and chain transmission 7

2.2 Post-quantum signature standardization: ML-DSA and SLH-DSA 8

2.3 Hybrid transition logic and deployment tension 8

2.4 Placement, exposure, and operational cost 8

3 Related Work **9**

3.1 Prior work on post-quantum TLS and hybrid handshakes 9

3.2 Prior work on certificate overhead and chain-related cost 9

3.3 Positioning of this paper 10

4 Research Questions and Study Scope **10**

4.1 Research questions 10

4.2 Scope and non-goals 11

5 Experimental Methodology **11**

5.1 Experimental objective 11

5.2 Implementation stack and execution environment 12

5.3 Scenario construction 12

5.4 Measurement model 13

5.5 Sample size, measurement stability, and interpretive scope 13

5.6 Experimental campaigns 13

5.7 Metric semantics and normalization 14

5.8 Methodological cautions 14

6 Results **15**

6.1 Global performance landscape 15

6.2 Campaign A: leaf-only comparison 16

6.2.1 Classical KEX . . . 17
6.2.2 Hybrid KEX . . . 17
6.2.3 Main result of Campaign A . . . 18
6.3 Campaign B: full hierarchy strategy matrix . . . 18
6.3.1 Fully-ML baseline . . . 18
6.3.2 Root-SLH with ML leaf . . . 19
6.3.3 Leaf-SLH strategies . . . 19
6.3.4 Main result of Campaign B . . . 20
6.4 Campaign C: depth comparison . . . 20
6.4.1 All-ML families . . . 20
6.4.2 Root-SLH with ML leaf . . . 21
6.4.3 Leaf-SLH families . . . 21
6.4.4 Main result of Campaign C . . . 22
6.5 Campaign D: KEX mode exploration . . . 22
6.5.1 Main result of Campaign D . . . 22

7 Cross-Cutting Cryptographic Interpretation 23
7.1 Signature placement versus mere signature presence . . . 23
7.2 Transport-related overhead versus cryptographic cost . . . 25
7.3 Effective chain exposure as a first-class variable . . . 26
7.4 Emergence of a distinct leaf-SLH regime . . . 27

8 Client/Server Cost Decomposition 28
8.1 Limits of client-only observations . . . 28
8.2 Balanced regime in all-ML scenarios . . . 29
8.3 Validation-skewed regime in upper-layer SLH scenarios . . . 30
8.4 Overwhelmingly server-bound regime in leaf-SLH scenarios . . . 31
8.5 Supporting microarchitectural signals . . . 32

9 Operational Implications for PQ TLS Deployment 33
9.1 From server-side handshake cost to service capacity . . . 33
9.2 Capacity loss under leaf-SLH . . . 34
9.3 Cost-per-million-handshakes interpretation . . . 35
9.4 Service-class translation . . . 36
9.5 Operational plausibility of hierarchy strategies . . . 37

10 Threats to Validity and Limitations 38

10.1 Local execution environment 38
10.2 Single implementation stack 38
10.3 Unequal run counts and scope of inference 39
10.4 Semantics of observed chain exposure 40
10.5 Perf as proxy rather than full function tracing 40
10.6 Economic model as derived interpretation 40

11 Conclusion **41**

11.1 Main finding 41
11.2 Implications for certificate hierarchy design 41
11.3 What remains open 42

A Scenario Inventory **44**

1 Introduction

1.1 Post-quantum TLS migration as protocol and PKI design

TLS 1.3 performs authenticated key establishment through an interactive protocol exchange in which certificate transmission, signature verification, and endpoint authentication contribute to handshake cost [11]. In certificate-based deployments, those costs depend on the negotiated key-establishment mechanism, the X.509 chain presented by the server, and the role of each certificate within that chain. Post-quantum migration in TLS therefore has to be studied at the level of the authenticated protocol path, where signature primitives are embedded in concrete certificate hierarchies and exercised under the timing constraints of the handshake [11, 12].

This perspective has gained relevance after the publication of NIST's first final post-quantum standards, including ML-KEM for key establishment, ML-DSA for digital signatures, and SLH-DSA as a stateless hash-based signature standard [5, 4, 6]. These standards define the primitive set considered here, but deployment choices inside latency-sensitive protocols remain protocol- and PKI-dependent. In TLS, signature families appear through certificate hierarchies, are transmitted in specific handshake messages, and are processed under interactive client/server constraints. The resulting migration problem belongs to protocol engineering and PKI design as well as to primitive selection.

The central design question in this paper concerns viability under placement. The issue is which post-quantum signature placements remain operationally plausible once TLS authentication is measured end to end, given a fixed protocol implementation and a concrete certification hierarchy. Under that reading, certificate-hierarchy design is part of the cryptographic migration problem.

1.2 Why signature placement matters

The cost of a signature family in TLS depends on its algorithmic properties and on its position in the certification hierarchy. A signature used in a long-lived trust anchor has a different protocol-visible effect from the same family used in the server leaf certificate presented directly in the interactive handshake. High-level discussions of post-quantum migration often obscure that distinction, even though placement is a primary deployment variable.

In TLS 1.3, the server certificate and transmitted chain participate directly in the authenticated portion of the handshake [11]. The leaf certificate is therefore a special position because it binds endpoint identity to the handshake and lies at the point where certificate design, verification cost, and live server-side authentication meet. Certificates placed higher in the hierarchy may still affect validation burden, chain size, and transmitted material, but their operational role differs from that of the leaf.

A placement-sensitive study of post-quantum authentication in TLS must therefore look beyond primitive-level benchmarks and certificate-size comparisons. Those measurements remain useful, but hierarchy-sensitive placement strategies also need direct evaluation, especially when a heavier signature family is either confined to upper trust layers or placed in the handshake-exposed server leaf.

1.3 Research gap

Prior work has shown that post-quantum authentication in TLS can introduce substantial overhead and that certificate-related effects matter in practice [12]. Other studies have explored mixed certificate chains as a transition strategy, making clear that heterogeneous hierarchies deserve direct analysis rather than incidental treatment as migration artifacts [9]. More recent

evaluation frameworks have expanded the measurement space by covering classical, hybrid, and pure post-quantum TLS configurations in controlled settings [3].

The placement-centered question remains less studied. Existing work has often emphasized primitive-level overhead, certificate-size growth, hybrid key exchange, or aggregate TLS performance under post-quantum integration. The position of a signature family within the certificate hierarchy has received comparatively less attention, even though it shapes the effective chain observed during the handshake, the burden visible in the live authentication path, and the distribution of cryptographic work across client and server.

Hierarchy position affects which certificates are exposed, which signatures are processed in the interactive path, and where the dominant cryptographic cost is paid. A transition study that ignores placement can flatten distinct operational regimes into a single configuration-level comparison.

1.4 Research questions and thesis

This paper is organized around the following research questions:

RQ1. To what extent is the operational cost of TLS 1.3 authentication determined by whether the server leaf certificate uses ML-DSA or SLH-DSA?

RQ2. Does placing SLH-DSA in upper layers of the certificate hierarchy behave differently from placing it in the leaf exposed to the interactive handshake?

RQ3. How do chain depth and effective chain transmission affect observed handshake latency and transferred data?

RQ4. To what extent is the observed degradation explained by transport size, and to what extent by cryptographic processing?

RQ5. Does moving from classical to hybrid or pure post-quantum key exchange materially change the main migration picture?

RQ6. What operational implications do these results have for organizations deploying interactive TLS services?

The paper advances a simple thesis. The dominant variable is not whether ML-DSA or SLH-DSA appears somewhere in the hierarchy, but where it appears. The main structural distinction separates scenarios in which SLH-DSA remains in upper trust layers from scenarios in which it reaches the server leaf exposed to the live TLS handshake.

The central claim follows from that distinction. Post-quantum TLS migration is best read as a certificate-hierarchy design problem. Signature placement, effective chain exposure, and client/server cryptographic burden jointly define different operational regimes, and the most severe regime is consistently associated with SLH-DSA in the interactive server leaf.

1.5 Contributions

1. The paper presents a hierarchy-sensitive experimental study of post-quantum TLS 1.3 authentication centered on certificate placement rather than flat primitive comparison.
2. It evaluates ML-DSA and SLH-DSA across four complementary experimental campaigns that isolate leaf placement, full hierarchy strategy, chain depth, and key-exchange mode.

3. It provides empirical evidence that signature placement explains the observed authentication regimes better than signature-family presence alone.
4. It shows that transport expansion does not by itself account for the dominant heavy regime observed in leaf-SLH scenarios, even though transport metrics remain highly informative outside that regime.
5. It demonstrates, through client/server performance decomposition, that leaf-SLH scenarios become overwhelmingly server-bound during live TLS authentication.
6. It releases the experimental code and scenario-generation material in a public repository to support traceability and reproducibility: https://github.com/hypergalois/PKI-PQC.
7. It translates the measured effects into operational terms, connecting post-quantum certificate-hierarchy design with deployment viability in interactive TLS services.

1.6 Paper organization

Section 2 introduces the cryptographic and protocol background needed to frame post-quantum authentication in TLS 1.3. Section 3 reviews the most relevant prior work and positions the present study within that literature. Section 4 states the research questions more formally and defines the scope of the study. Section 5 presents the experimental methodology, including the implementation stack, scenario construction, measurement model, sampling strategy, and metric semantics. Section 6 reports the campaign-level results. Section 7 develops a cross-cutting cryptographic interpretation of those results, with particular attention to signature placement, chain exposure, and the distinction between transport-related and cryptographic cost. Section 8 analyzes client/server workload decomposition. Section 9 translates the measured effects into operational implications for post-quantum TLS deployment. Section 10 discusses threats to validity and limitations. Section 11 concludes.

2 Cryptographic and Protocol Background

2.1 TLS 1.3 certificate authentication and chain transmission

TLS 1.3 provides authenticated key establishment by combining ephemeral key exchange with transcript-bound authentication [11]. In the certificate-based server path, the server sends a certificate chain in the `Certificate` message and proves possession of the corresponding private key in `CertificateVerify`. Both messages are bound to the handshake transcript, so certificate handling belongs to the live cryptographic execution of the protocol and its cost is observed during the handshake.

In Internet deployments, the certificates used along that path typically follow the X.509 profile standardized for PKIX, with a leaf certificate identifying the authenticated endpoint and additional certificates carrying the issuing chain required for validation [1]. TLS authentication cost therefore depends on the negotiated key-establishment mechanism, the structure and contents of the certificate path presented during the handshake, and the position of each certificate in that path. The server leaf has a special role because it is tied directly to endpoint identity and to the key used in the interactive authentication step.

TLS authenticates a concrete endpoint through a concrete certificate path transmitted under protocol timing constraints. Certificate hierarchy, effective chain exposure, and verification burden can therefore affect the observed behavior of the handshake. In the post-quantum setting,

where signature families may differ sharply in size and computational profile, those effects become more visible [12, 9].

2.2 Post-quantum signature standardization: ML-DSA and SLH-DSA

NIST's first final post-quantum standards define ML-KEM for key establishment, ML-DSA for digital signatures, and SLH-DSA as a stateless hash-based digital signature standard [5, 4, 6]. For TLS authentication, ML-DSA and SLH-DSA are standardized signature options with different design foundations within post-quantum cryptography.

ML-DSA is a module-lattice-based signature standard derived from the CRYSTALS-Dilithium lineage and intended as a general-purpose digital signature mechanism for the post-quantum transition [4]. SLH-DSA is a stateless hash-based standard derived from the SPHINCS+ family and reflects a different set of cryptographic trade-offs grounded in hash-based security assumptions and stateless tree-based signing [6]. Both are standardized options, but standardization does not imply similar behavior once they are embedded in certificate hierarchies and exercised inside a live TLS handshake.

The present paper keeps that distinction at the level relevant to the protocol setting under study. It does not attempt a full algorithmic comparison of ML-DSA and SLH-DSA across all possible uses. Its narrower concern is TLS 1.3 certificate authentication, where signature families are consumed through transmitted and validated certification paths. In that setting, standardization establishes cryptographic legitimacy, but not operational equivalence.

2.3 Hybrid transition logic and deployment tension

The transition to post-quantum cryptography is widely understood as a migration problem rather than a one-step replacement event. In TLS, this has been especially visible on the key-establishment side, where hybrid designs combine classical and post-quantum components so that security is preserved if at least one constituent remains secure during the transition period [13, 2]. That logic is motivated by compatibility, cryptographic conservatism, and the practical need to move incrementally instead of assuming immediate ecosystem-wide convergence [10].

A related tension appears in certificate-based authentication. Organizations migrate certificate hierarchies under trust-anchor lifetimes, interoperability constraints, implementation limits, validation behavior, and operational latency budgets. In TLS, the post-quantum transition is therefore tied to the choice of signature primitives, to how those primitives are distributed across the certification hierarchy, and to which costs are exposed in the live handshake [9, 12].

Mixed and hierarchy-sensitive strategies address that deployment setting directly. A deployment may place a heavier or more conservative signature family in upper trust layers while preserving a different family at the interactive server leaf, or it may choose the reverse placement. Those choices are part of transition design because they determine how authentication cost appears during protocol execution.

2.4 Placement, exposure, and operational cost

The central premise of this paper is that certificate placement within the hierarchy is a primary variable in post-quantum TLS authentication. The relevant issue is which signature family appears in the chain, which certificates are exposed during the handshake, and where the corresponding cryptographic burden is paid. Prior work on post-quantum authentication in TLS and on mixed certificate chains suggests that certificate-related effects can dominate practical overhead and that heterogeneous hierarchies require explicit treatment during the transition [12, 9].

That point becomes sharper once logical hierarchy is separated from effective chain exposure. A PKI may contain roots, intermediates, and leaves as part of its declared structure, but the set of certificates actually transmitted and processed during the handshake is an empirical property of the deployed authentication path. Effective exposure influences transmitted bytes, validation burden, and potentially the concentration of active work across client and server roles. It is therefore part of the cryptographic cost surface of the protocol rather than a secondary representational detail.

Under that view, placement has direct operational meaning. A signature family used in an upper trust layer may affect path validation and chain size while leaving the live server-authentication step within a bounded cost regime. The same family, once moved into the handshake-exposed leaf, may alter the execution profile of the entire authentication path. The present study examines that distinction systematically by treating hierarchy position, effective chain exposure, and client/server cost concentration as joint variables in post-quantum TLS 1.3 authentication.

3 Related Work

3.1 Prior work on post-quantum TLS and hybrid handshakes

Prior work has established TLS as a main protocol setting for empirical evaluation of post-quantum migration beyond primitive-level analysis. Sikeridis, Kampanakis, and Devetsikiotis studied post-quantum authentication in TLS 1.3 and showed that signature choice can materially affect connection-establishment latency and throughput under realistic deployment assumptions [12]. Their results quantify overhead in the authentication path and show that this path is a relevant deployment surface for post-quantum cryptography.

Subsequent work expanded the transition picture by examining mixed and hybrid configurations. In the authentication space, Paul et al. proposed mixed certificate chains for TLS 1.3 as an explicit migration strategy, arguing that different signature algorithms may be assigned to different positions in the same hierarchy to balance trust, interoperability, and performance [9]. On the key-establishment side, current IETF work on hybrid key exchange and ML-KEM deployment in TLS follows a related transition model, in which compatibility and security margin have to coexist during staged deployment [13, 2].

The practical experimentation ecosystem has also matured. The Open Quantum Safe project is a common platform for prototyping post-quantum cryptography in real protocol stacks, including OpenSSL-based TLS deployments through liboqs and oqsprovider [7, 8]. More recent measurement-oriented work, such as the framework of Montenegro et al., has expanded the empirical basis for post-quantum TLS evaluation by enabling comparative study across classical, hybrid, and pure post-quantum configurations in a unified environment [3]. Taken together, these lines of work define PQ TLS as a concrete domain of empirical cryptographic research.

3.2 Prior work on certificate overhead and chain-related cost

Within that literature, certificate-related overhead has emerged as a major source of deployment cost. Sikeridis et al. showed that post-quantum authentication in TLS 1.3 cannot be evaluated solely through abstract security arguments or primitive-level operation counts, because certificate size, transmitted bytes, and signature verification cost all affect handshake behavior materially [12]. Their study shifted the discussion from purely algorithmic comparison toward protocol-visible authentication overhead.

Paul et al. extended that line of analysis by treating the certificate chain itself as a design space instead of a fixed administrative artifact [9]. Their mixed-chain proposal is especially relevant because it recognizes that post-quantum migration may proceed through heterogeneous hierarchies in which different certificates serve different operational roles. It suggests that the effective cost of post-quantum authentication depends on algorithm selection and on the distribution of those algorithms across the hierarchy.

Recent PQ TLS evaluation frameworks have continued to track the relation between cryptographic choices, handshake latency, and protocol overhead [3]. Even so, prior work has generally emphasized aggregate configuration-level behavior more than placement-centered analysis of certification hierarchies. The literature has shown that certificates matter. It has been less explicit about how hierarchy position, effective chain exposure, and client/server cost concentration interact as distinct explanatory variables.

3.3 Positioning of this paper

The present study addresses that placement-centered gap. It focuses on the certificate hierarchy itself: which signature family is placed at the root, which at the intermediate, which at the server leaf, and how that placement shapes the cost of live TLS authentication. The study includes classical, hybrid, and pure post-quantum KEX modes in the experimental design, but its main object is the authentication hierarchy rather than key-exchange substitution or uniform signature replacement.

This choice distinguishes the present work from prior evaluation efforts in two ways. First, hierarchy-sensitive signature placement is treated as the main explanatory variable rather than as a secondary parameter inside a broader benchmark. Second, hierarchy placement is analyzed together with effective chain exposure and client/server workload decomposition during the handshake. The paper studies post-quantum TLS authentication as a problem of certificate-path construction and exposure, as well as aggregate overhead [12, 9, 3].

The paper asks which certificate-chain strategies remain operationally plausible once post-quantum signature placement is evaluated inside the authenticated handshake path of TLS 1.3.

4 Research Questions and Study Scope

4.1 Research questions

The study is organized around six research questions:

RQ1. To what extent is the operational cost of TLS 1.3 certificate-based authentication determined by whether the server leaf certificate uses ML-DSA or SLH-DSA?

RQ2. Does placing SLH-DSA in upper layers of the certificate hierarchy behave materially differently from placing it in the handshake-exposed server leaf?

RQ3. How do hierarchy depth and effective chain exposure shape observed handshake latency and transmitted data during TLS 1.3 authentication?

RQ4. To what extent is the observed degradation explained by transport-related overhead, and to what extent by cryptographic processing cost?

RQ5. Does moving from classical to hybrid or pure post-quantum key establishment materially alter the main migration picture?

RQ6. What operational implications do these results have for organizations deploying interactive TLS services under post-quantum transition constraints?

Taken together, these questions define the analytical focus of the paper. The study determines how certificate-hierarchy design shapes the cost of live TLS 1.3 authentication once signature placement, chain exposure, and client/server burden are considered jointly.

4.2 Scope and non-goals

The paper studies a concrete cryptographic and operational question: how post-quantum signature placement within TLS 1.3 certificate hierarchies affects the behavior of live certificate-based authentication in an implementation stack built on OpenSSL 3 and oqsprovider [11, 7, 8]. The aim is controlled evaluation of a specific migration problem with practical relevance, not universal prediction across all TLS libraries, PKI deployments, or Internet environments.

The scope is bounded in several ways. First, the evaluation targets one implementation stack. Absolute values may depend on library internals, provider integration, certificate handling behavior, and platform-specific factors. The strongest claims advanced here are structural: hierarchy-sensitive signature placement can decisively shape the operational regime of post-quantum TLS authentication.

Second, the analysis does not include full internal function tracing or exhaustive microarchitectural attribution. Performance counters are used as regime-level evidence for client/server cost concentration, not as a replacement for complete code-path reconstruction or low-level implementation forensics.

Third, the study does not measure Internet-wide or WAN-visible latency under arbitrary production conditions. The experiments are conducted in a controlled local environment so that certificate-path effects, chain exposure, and cryptographic cost can be observed with reduced network noise. The reported latency values should therefore be read as comparative measurements within a real stack, not as direct forecasts of user-perceived latency on the public Internet.

Finally, the study does not settle which post-quantum primitive is best across all protocol settings. It addresses a narrower question: which certificate-chain strategies remain operationally plausible for TLS 1.3 server authentication when post-quantum signatures are embedded into live X.509-based authentication paths [11, 1, 4, 6].

5 Experimental Methodology

5.1 Experimental objective

The experimental objective is to measure how the placement of ML-DSA and SLH-DSA within X.509 certification paths affects the cost of live server authentication during the TLS handshake [11, 1, 4, 6].

The laboratory evaluates hierarchy-sensitive authentication scenarios at the protocol level. The study asks which hierarchy constructions remain operationally plausible once authentication is exercised through a concrete certificate path, transmitted during a real TLS 1.3 handshake, and measured as an end-to-end cryptographic event. This framing is consistent with prior work showing that post-quantum authentication overhead in TLS is shaped by certificate-related effects and is best studied at the protocol level instead of being inferred from primitive properties alone [12, 9, 3].

5.2 Implementation stack and execution environment

All experiments were conducted in a local TLS laboratory built around OpenSSL 3 and the Open Quantum Safe provider stack. The cryptographic implementation used oqsprovider on top of liboqs, which enabled the post-quantum signature and key-establishment mechanisms required for the study [7, 8]. The signature families of interest were ML-DSA and SLH-DSA, instantiated through concrete algorithm selections appropriate to the chosen security level. On the key-establishment side, the experimental design considered three modes: classical `X25519`, hybrid `X25519MLKEM768`, and pure post-quantum `MLKEM768` [5, 4, 6, 2, 13].

The server side was driven through `openssl s_server`, parameterized per scenario to load the corresponding certificate chain, key material, and TLS group configuration. The client side used a dedicated benchmark client written in C and instrumented for per-handshake measurement. For each run, the client established a fresh TCP connection, forced TLS 1.3, loaded the required provider configuration, selected the key-establishment group for the scenario under test, and executed a full handshake against the server. The unit of observation is therefore the complete certificate-based authentication event, excluding resumed sessions and amortized multi-request connection artifacts.

The code used to generate certificates, construct scenarios, drive measurements, and organize the experimental artifacts is available in the public repository https://github.com/hypergalois/PKI-PQC.

5.3 Scenario construction

The scenario space was designed to isolate hierarchy-sensitive authentication effects. Each scenario is defined by a combination of certificate-hierarchy depth, signature-family placement within the hierarchy, and key-establishment mode.

Hierarchy depth was varied between depth 2 and depth 3 configurations. In depth 2 scenarios, the logical certification path consisted of a root and a leaf. In depth 3 scenarios, an intermediate certificate was inserted between root and leaf. This distinction was introduced to examine how hierarchy depth interacts with effective chain exposure and with the placement of heavier signature families.

Within each hierarchy, signature families were assigned positionally. The main placement variables are therefore root, intermediate, and leaf. This positional model distinguishes scenarios in which SLH-DSA appears only in upper trust layers, scenarios in which it appears directly in the server leaf, and scenarios in which the entire hierarchy uses a common family. The design supports the question of whether the same signature family has different operational meaning depending on where it is embedded in the certification path.

The experimental matrix included both uniform and mixed hierarchies. Uniform hierarchies were used to define the all-ML baseline and selected all-SLH comparison points. Mixed hierarchies were used to evaluate transitional strategies such as root-SLH with ML-DSA preserved in the leaf, as well as heavier placements in which SLH-DSA reaches the interactive certificate. The key-establishment dimension was varied across classical, hybrid, and pure post-quantum modes to determine whether the placement-sensitive picture changes under different KEX assumptions.

For analysis purposes, the study also defines comparable families of scenarios. These groups differ along one principal axis while preserving the rest of the hierarchy structure as far as possible. Examples include leaf-only contrasts under fixed surrounding conditions, depth 2 versus depth 3 comparisons within the same high-level family, and KEX comparisons under comparable chain

constructions. Comparability supports causal interpretation because the paper is concerned less with aggregate scenario ranking than with identifying which structural variable governs each observed regime.

5.4 Measurement model

The measurement model is defined at the level of individual handshakes. Each experimental run establishes a fresh TCP connection, performs a TLS 1.3 handshake under the selected scenario, and records protocol-visible and system-visible metrics. The per-handshake design uses live certificate-based authentication as the basic unit of analysis.

Handshake latency was measured using a monotonic clock source to avoid wall-clock artifacts and preserve a stable basis for repeated comparisons. At the transport level, the client recorded the number of bytes read and written during the handshake. At the certificate-path level, it recorded the observed chain length and the observed chain size in DER bytes, using the peer certificate chain exposed by the TLS stack together with certificate serialization. A defensive fallback to the leaf certificate was maintained so that the measurement pipeline continued to produce data when chain exposure did not match a naive logical expectation.

In addition to protocol-level metrics, the experiments collected performance-counter data on both client and server. These counters were normalized per run to support client/server decomposition at the handshake level. The principal performance-counter variables used in the paper are task-clock, instructions, cycles, and derived ratios such as task-clock over elapsed time and server/client task-clock ratio. These variables indicate whether a given regime remains balanced, becomes validation-skewed, or becomes overwhelmingly server-bound.

5.5 Sample size, measurement stability, and interpretive scope

Run counts were not uniform across scenarios. Fast scenarios were executed with larger sample sizes, whereas extremely heavy scenarios, especially those involving SLH-DSA in the leaf, were run with smaller but still analytically sufficient counts. When one class of scenarios completes in well under a millisecond and another takes on the order of seconds, enforcing identical run counts across the entire matrix would spend disproportionate wall-clock budget on scenarios whose regime is already sharply separated from the rest.

This choice should be read in light of the inferential aim of the paper. The study is not trying to resolve tiny marginal differences among near-equivalent configurations. Its main objective is to identify regime changes, placement-sensitive discontinuities, and shifts in where cryptographic work is concentrated. Under that objective, a smaller number of runs in a seconds-scale scenario does not carry the same interpretive risk that it would in a study focused on sub-percent differences between tightly clustered alternatives.

The non-uniform sampling policy balances feasibility and descriptive stability. Fast scenarios were run more extensively because they were inexpensive to execute and because denser repetition is useful when characterizing low-latency regions. Heavy leaf-SLH scenarios were run fewer times because each run was costly, yet the observed differences remained large, consistent, and visible across multiple metrics, including mean latency, p95 latency, transport volume, and server-side task-clock.

5.6 Experimental campaigns

The experimental design is organized into four campaigns, each isolating a distinct question in post-quantum TLS authentication.

Campaign A isolates the leaf effect. It compares scenarios in which the principal change is the signature family used in the server leaf, while keeping the surrounding hierarchy as simple and controlled as possible. Its purpose is to test whether the main discontinuity already appears in the most direct certificate-authentication comparison.

Campaign B studies the full hierarchy strategy space under a common hybrid key-establishment regime. This is the central strategy matrix of the paper. It evaluates complete hierarchy constructions rather than local pairwise substitutions, allowing the study to compare migration strategies as complete authentication designs.

Campaign C isolates topology from placement by comparing depth 2 and depth 3 hierarchies across comparable algorithmic families. Its purpose is to determine whether increasing chain depth behaves as a simple monotonic penalty or whether the effect depends on which certificates become visible in the effective chain transmitted during the handshake.

Campaign D explores key-establishment variation under comparable chain constructions. It contrasts classical, hybrid, and pure post-quantum KEX modes to determine whether the main placement-driven interpretation survives across different key-establishment assumptions.

5.7 Metric semantics and normalization

The paper uses a layered metric model. At the handshake level, the main latency variables are `elapsed_mean_ms` and `elapsed_p95_ms`, which capture average and high-percentile latency, respectively. These are the principal observables used to assess interactive plausibility in live TLS authentication.

At the transport and chain level, the main variables are `bytes_read_mean`, `bytes_written_mean`, `chain_bytes_unique`, and `served_chain_der_bytes`. The first two track protocol-visible byte cost during the handshake. The latter two characterize certificate material exposure more directly, although their semantics are not perfectly homogeneous across all topology classes, as discussed below.

At the performance-counter level, the main variables are `client_task_clock_per_run_ms`, `server_task_clock_per_run_ms`, instructions, cycles, IPC, and derived normalized ratios. These variables allow the analysis to distinguish among balanced low-cost regimes, client-skewed validation regimes, and overwhelmingly server-bound regimes.

Several metrics are also expressed relative to a common baseline. The baseline scenario used throughout the paper is the hybrid depth 3 fully-ML hierarchy:

`x25519mlkem768__ml_root__ml_int__ml_leaf`.

Relative metrics such as latency multiplier, bytes-read multiplier, retained capacity, and server-side compute multiplier make structural contrasts easier to read than raw absolute values alone. This normalization is especially useful because the experimental space contains both sub-millisecond and second-scale regimes.

Finally, the paper derives capacity and economic interpretations from normalized server-side compute measurements. These derived quantities are not treated as direct billing observations, but as operational translations of per-handshake server burden.

5.8 Methodological cautions

Several methodological cautions are necessary for correct interpretation of the dataset.

Table 1: Master scenario summary (short version).

scenario_id	kex_mode	depth	hierarchy	mean_ms	p95_ms	bytes_read	server_task_ms	srv_cli_ratio
x25519__leaf_mldsa65	classical	2	ML root / ML leaf	0.688	0.874	14904	0.529	1.022
x25519__leaf_slhdsashake192s	classical	2	SLH root / SLH leaf	1464.933	1529.999	49881	1462.281	462.869
x25519mlkem768__leaf_mldsa65	hybrid	2	ML root / ML leaf	0.841	1.072	15992	0.623	1.049
x25519mlkem768__leaf_slhdsashake192s	hybrid	2	SLH root / SLH leaf	1413.991	1448.787	50969	1411.345	448.264
x25519mlkem768__ml_root__ml_leaf	hybrid	2	ML root / ML leaf	0.809	1.013	15992	0.602	1.044
x25519mlkem768__ml_root__slh_leaf	hybrid	2	ML root / SLH leaf	1406.283	1438.054	26999	1404.927	696.186
x25519mlkem768__slh_root__ml_leaf	hybrid	2	SLH root / ML leaf	3.376	3.856	39962	1.915	0.998
x25519mlkem768__slh_root__slh_leaf	hybrid	2	SLH root / SLH leaf	1407.714	1426.652	50969	1405.087	441.560
x25519mlkem768__ml_root__ml_int__ml_leaf	hybrid	3	ML root / ML int / ML leaf	0.809	1.000	16008	0.562	0.903
x25519mlkem768__ml_root__ml_int__slh_leaf	hybrid	3	ML root / ML int / SLH leaf	1402.486	1408.711	27015	1401.169	687.949
x25519mlkem768__ml_root__slh_int__slh_leaf	hybrid	3	ML root / SLH int / SLH leaf	1405.803	1419.470	38046	1403.109	433.616
x25519mlkem768__slh_root__ml_int__ml_leaf	hybrid	3	SLH root / ML int / ML leaf	2.133	2.522	28947	0.667	0.346
x25519mlkem768__slh_root__ml_int__slh_leaf	hybrid	3	SLH root / ML int / SLH leaf	1403.166	1409.142	39954	1401.849	430.764
x25519mlkem768__slh_root__slh_int__slh_leaf	hybrid	3	SLH root / SLH int / SLH leaf	1408.703	1423.678	50985	1404.903	326.456
mlkem768__ml_root__ml_leaf	pure_pqc	2	ML root / ML leaf	0.665	0.887	15960	0.515	1.003
mlkem768__slh_root__slh_leaf	pure_pqc	2	SLH root / SLH leaf	1405.456	1414.012	50937	1402.896	463.910
mlkem768__slh_root__ml_int__ml_leaf	pure_pqc	3	SLH root / ML int / ML leaf	1.884	2.186	28915	0.527	0.298

First, the analytical dataset used for cross-cutting interpretation is deduplicated by `scenario_id`. Campaign C intentionally reuses scenarios that also appear in Campaign B. These repeated scenarios were checked for metric consistency and treated as analytical reuse rather than as independent new observations. Campaign identity is preserved where narratively relevant, but scenario-level cross-cutting analysis operates on the deduplicated space.

Second, the observed value `chain_len_unique = 2` does not imply that the logical certification hierarchy has depth 2. In the dataset studied here, the client consistently observes two certificates in the effective chain, but those two certificates do not always correspond to the same logical pair. In depth 2 scenarios, the observed pair may correspond to root plus leaf; in depth 3 scenarios, it may instead correspond to intermediate plus leaf. Effective chain exposure must therefore be treated as an empirical variable rather than inferred mechanically from declared hierarchy depth.

Third, `served_chain_der_bytes` does not carry identical semantics across all topology classes. In the present dataset, it aligns with leaf DER size in some depth 2 scenarios, whereas in depth 3 scenarios it more closely tracks the observed effective chain. For strict transport reasoning across depths, the more reliable observables are therefore `bytes_read_mean` and `chain_bytes_unique`.

Fourth, the performance-counter analysis is regime-oriented. The paper uses perf-derived data to identify where active work concentrates during the handshake, not to provide complete code-path attribution. Strong claims are therefore made at the level of cost concentration and regime structure, rather than at the level of unique internal implementation causes.

6 Results

6.1 Global performance landscape

The global results separate into distinct operational regimes rather than a smooth continuum of gradually increasing cost. All-ML scenarios remain in a narrow low-latency band below one millisecond, whereas scenarios with SLH-DSA in the server leaf cluster around a plateau near 1.4 seconds. Between these regions lies a smaller intermediate band, populated mainly by configurations that place SLH-DSA in upper hierarchy layers while preserving ML-DSA in the interactive leaf.

Figure 1 shows mean latency ranging from approximately 0.665 ms to 1464.933 ms. This spread reflects regime separation rather than a modest gradient of penalties. The low-cost region contains all-ML scenarios and a small number of root-SLH / leaf-ML cases, while the heavy region is dominated by configurations in which SLH-DSA reaches the leaf.

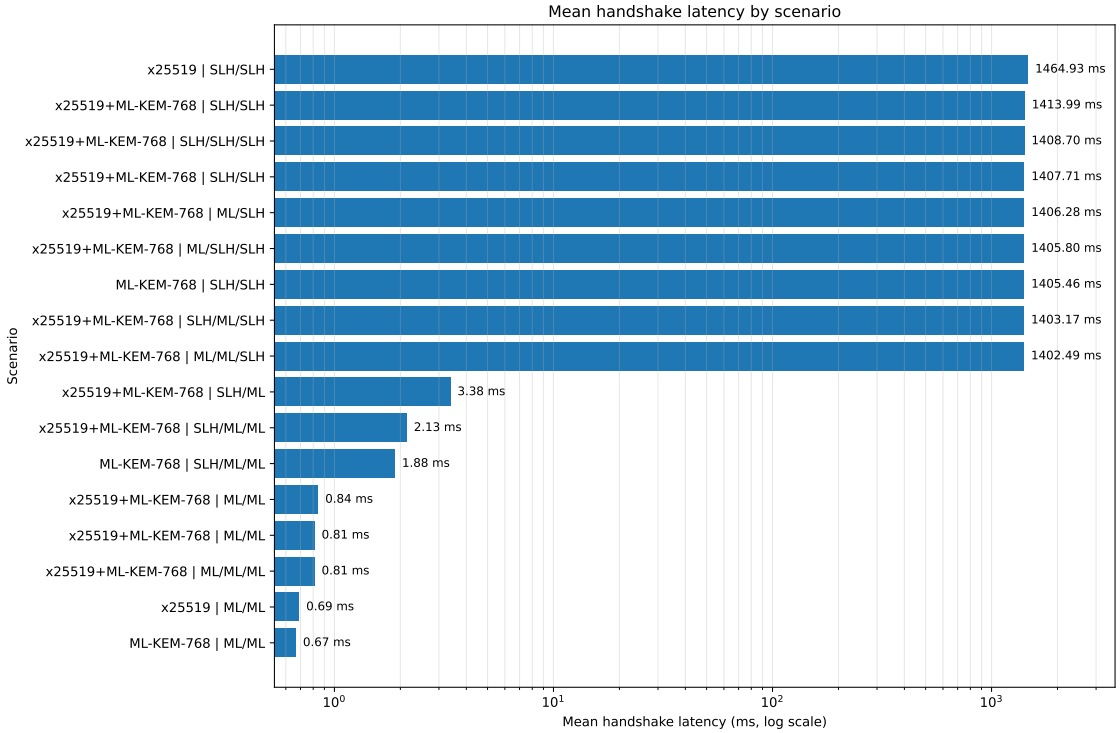


Figure 1: Mean handshake latency by scenario.

Table 2: Campaign A paired comparison.

kex_mode	tls_group_family	ml elapsed mean (ms)	slh elapsed mean (ms)	latency ratio SLH / ML	ml bytes read mean	slh bytes read mean	bytes-read ratio SLH / ML
classical	x25519	0.688	1464.933	2127.865	14904	49881	3.347
hybrid	x25519mlkem768	0.841	1413.991	1682.137	15992	50969	3.187

kex_mode	tls_group_family	ml server task clock per run (ms)	slh server task clock per run (ms)	server taskclock ratio SLH / ML	ml client task clock per run (ms)	slh client task clock per run (ms)	client taskclock ratio SLH / ML
classical	x25519	0.529	1462.281	2765.628	0.517	3.159	6.107
hybrid	x25519mlkem768	0.623	1411.345	2265.961	0.594	3.148	5.303

Figure 2 shows the same structure in high-percentile behavior. The heavy scenarios shift upward in both mean and upper-tail latency, indicating that the severe regime is a persistent property of the handshake rather than a minor average-case distortion.

The heavy scenarios do not follow a broad gradient in which progressively larger chains produce progressively worse latency. Once SLH-DSA is placed in the leaf, they cluster into a relatively tight plateau, which points to placement as the main discontinuity rather than to incremental transport growth alone.

6.2 Campaign A: leaf-only comparison

Campaign A isolates the leaf effect while keeping the surrounding hierarchy simple and controlled. It tests whether the main discontinuity appears as soon as SLH-DSA is introduced into the interactive server leaf.

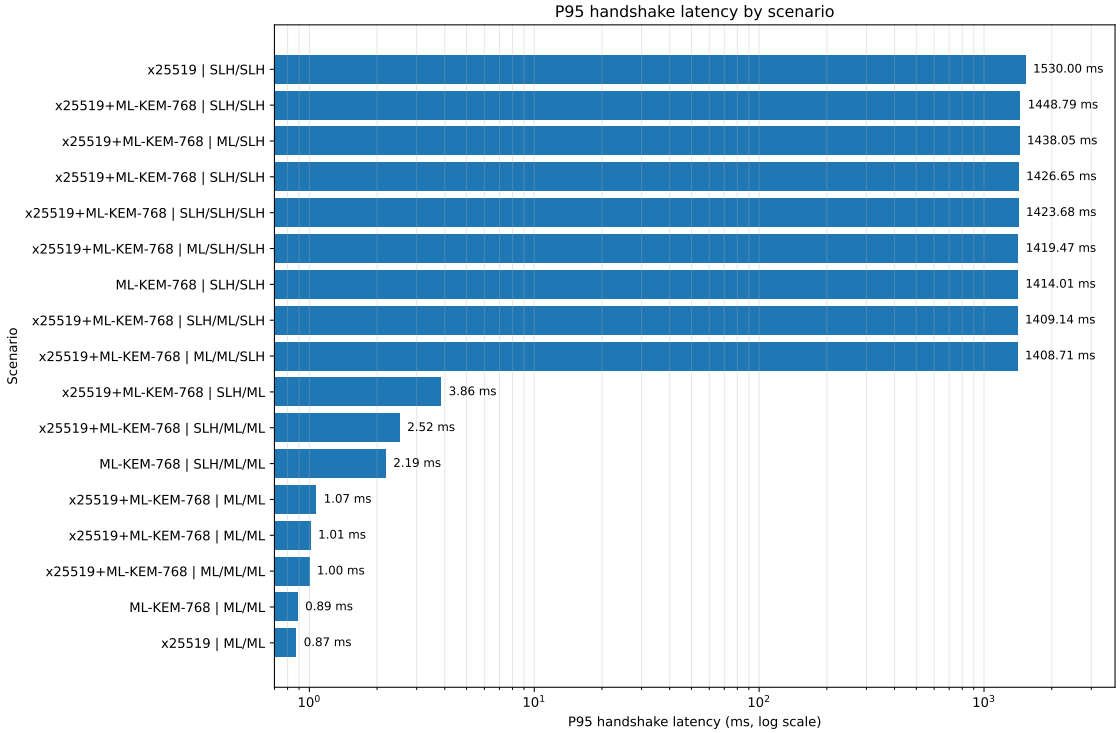


Figure 2: P95 handshake latency by scenario.

6.2.1 Classical KEX

Under classical `X25519`, replacing an ML-DSA leaf with an SLH-DSA leaf produces an extreme discontinuity. Mean latency increases by approximately 2127.86×, and p95 by approximately 1750.57×. The transport expansion is much smaller: `bytes_read_mean` grows by about 3.35×, and `chain_bytes_unique` by about 2.98×.

The same asymmetry appears in the performance counters. Server task-clock per run grows by approximately 2765.63×, whereas client task-clock per run grows by only 6.11×. Even in this simplest leaf-only configuration, the latency jump is far larger than the increase in transmitted certificate material.

6.2.2 Hybrid KEX

The same comparison under hybrid `X25519MLKEM768` yields the same qualitative result. Mean latency increases by approximately 1682.14×, while p95 rises by approximately 1351.48×. Transport growth again remains comparatively modest: `bytes_read_mean` grows by about 3.19×, and `chain_bytes_unique` by about 2.98×.

The performance counters support the same interpretation. Server task-clock per run increases by approximately 2265.96×, while client task-clock per run increases by only 5.30×. The heavy regime is therefore visible already in the simplest leaf-only comparison and does not depend on deeper hierarchies or more elaborate mixed chains.

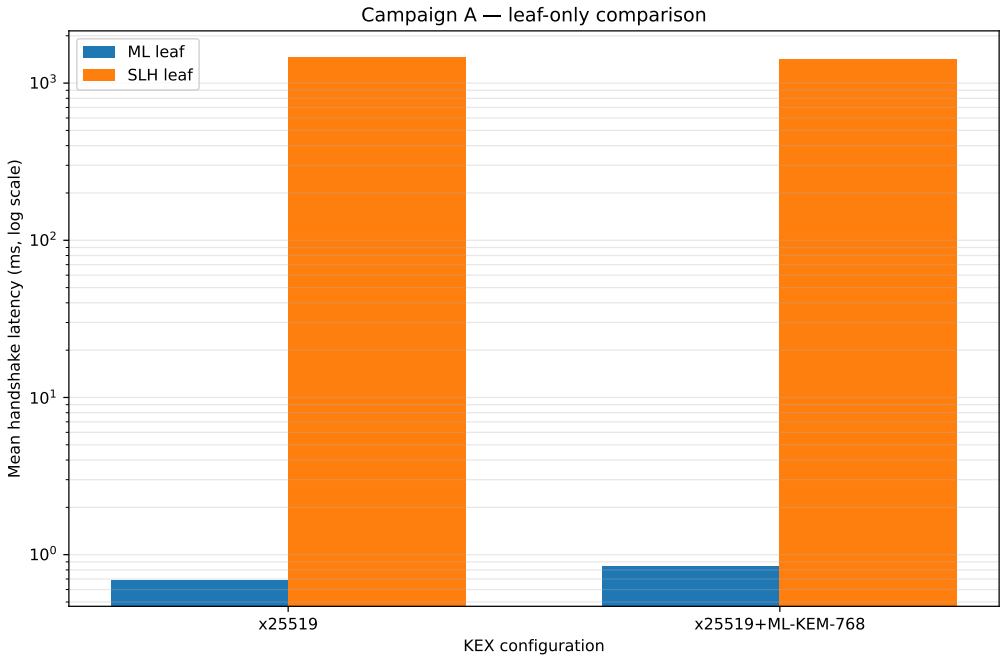


Figure 3: Campaign A: leaf-only comparison.

Table 3: Campaign B strategy matrix.

scenario_id	slh_position_class	elapsed_mean_ms	bytes_read_mean	server_task_clock_per_run_ms	latency_relative_to_baseline
x25519mlkem768__ml_root__ml_int__ml_leaf	no_slh	0.809000	16008	0.562000	1.000000
x25519mlkem768__slh_root__ml_int__ml_leaf	root	2.133000	28947	0.667000	2.640000
x25519mlkem768__ml_root__ml_int__slh_leaf	leaf	1402.486000	27015	1401.169000	1733.490000
x25519mlkem768__slh_root__ml_int__slh_leaf	root_and_leaf	1403.166000	39954	1401.849000	1734.330000
x25519mlkem768__ml_root__slh_int__slh_leaf	intermediate_and_leaf	1405.803000	38046	1403.109000	1737.590000
x25519mlkem768__slh_root__slh_int__slh_leaf	root_and_intermediate_and_leaf	1408.703000	50985	1404.903000	1741.180000

scenario_id	slh_position_class	server_cpu_relative_to_baseline	bytes_read_relative_to_baseline	operational_plausibility
x25519mlkem768__ml_root__ml_int__ml_leaf	no_slh	1.000000	1.000000	Reasonable
x25519mlkem768__slh_root__ml_int__ml_leaf	root	1.190000	1.810000	Penalized but plausible
x25519mlkem768__ml_root__ml_int__slh_leaf	leaf	2493.640000	1.690000	Unsuitable for interactive TLS front-end
x25519mlkem768__slh_root__ml_int__slh_leaf	root_and_leaf	2494.850000	2.500000	Unsuitable for interactive TLS front-end
x25519mlkem768__ml_root__slh_int__slh_leaf	intermediate_and_leaf	2497.090000	2.380000	Unsuitable for interactive TLS front-end
x25519mlkem768__slh_root__slh_int__slh_leaf	root_and_intermediate_and_leaf	2500.280000	3.180000	Unsuitable for interactive TLS front-end

6.2.3 Main result of Campaign A

Campaign A shows that the dominant operational discontinuity does not require a complex hierarchy, a mixed certification strategy, or a pure post-quantum key-establishment path. It appears as soon as SLH-DSA is moved into the server leaf certificate, before the analysis reaches the full hierarchy design space.

6.3 Campaign B: full hierarchy strategy matrix

Campaign B evaluates complete hierarchy constructions under a common hybrid key-establishment regime. This campaign compares migration strategies as full authentication designs rather than as local substitutions.

6.3.1 Fully-ML baseline

The baseline strategy for Campaign B is the hybrid depth-3 hierarchy `x25519mlkem768__ml_root__ml_int__ml_leaf`. Its mean latency is 0.809 ms, its p95 is 1.000 ms, it reads 16008 bytes per handshake on average, and its server task-clock per run is approximately

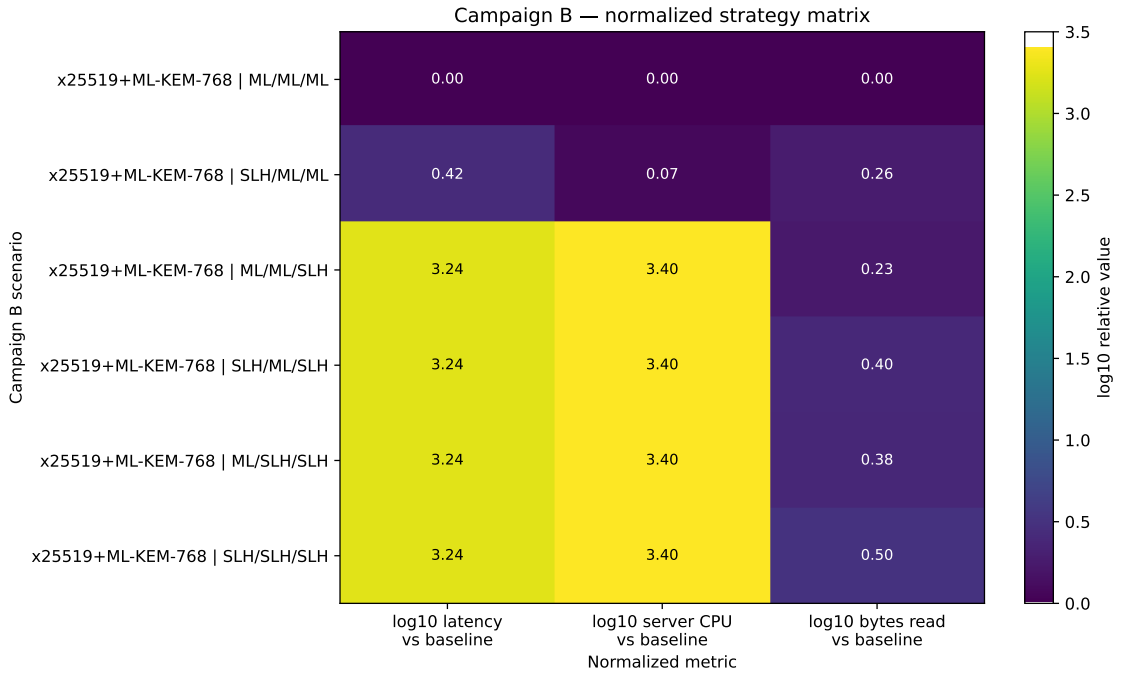


Figure 4: Campaign B: normalized strategy matrix.

Table 4: Campaign B operational plausibility ranking.

plausibility_rank	scenario_id	hierarchy_family_label	slh_position_class	elapsed_mean_ms	server_task_clock_per_run_ms	operational_plausibility
1	x25519mlkem768__ml_root__ml_int__ml_leaf	ML root / ML int / ML leaf	no_slh	0.809000	0.562000	Reasonable
2	x25519mlkem768__slh_root__ml_int__ml_leaf	SLH root / ML int / ML leaf	root	2.133000	0.667000	Penalized but plausible
4	x25519mlkem768__ml_root__ml_int__slh_leaf	ML root / ML int / SLH leaf	leaf	1402.486000	1401.169000	Unsuitable for interactive TLS front-end
4	x25519mlkem768__slh_root__ml_int__slh_leaf	SLH root / ML int / SLH leaf	root_and_leaf	1403.166000	1401.849000	Unsuitable for interactive TLS front-end
4	x25519mlkem768__ml_root__slh_int__slh_leaf	ML root / SLH int / SLH leaf	intermediate_and_leaf	1405.803000	1403.109000	Unsuitable for interactive TLS front-end
4	x25519mlkem768__slh_root__slh_int__slh_leaf	SLH root / SLH int / SLH leaf	root_and_intermediate_and_leaf	1408.703000	1404.903000	Unsuitable for interactive TLS front-end

0.562 ms. This scenario is the reference point for the relative comparisons reported throughout the paper.

6.3.2 Root-SLH with ML leaf

The most important bounded-penalty case in Campaign B is `x25519mlkem768__slh_root__ml_int__ml_leaf`. Relative to the fully-ML baseline, its mean latency rises by approximately 2.64×, its server task-clock increases by about 1.19×, and its read volume increases by about 1.81×. The scenario remains within an interactive regime and is the only non-baseline strategy in Campaign B that remains defensible under a strict operational reading.

6.3.3 Leaf-SLH strategies

The remaining mixed strategies in Campaign B all place SLH-DSA in the leaf. Once that occurs, the design space contracts into a narrow high-cost band. Mean latency ranges from approximately 1402.486 ms to 1408.703 ms, corresponding to roughly 1733.49× to 1741.18× the fully-ML baseline. Over the same range, server task-clock grows by approximately 2493.64× to 2500.28×, while read volume grows by only about 1.69× to 3.18×.

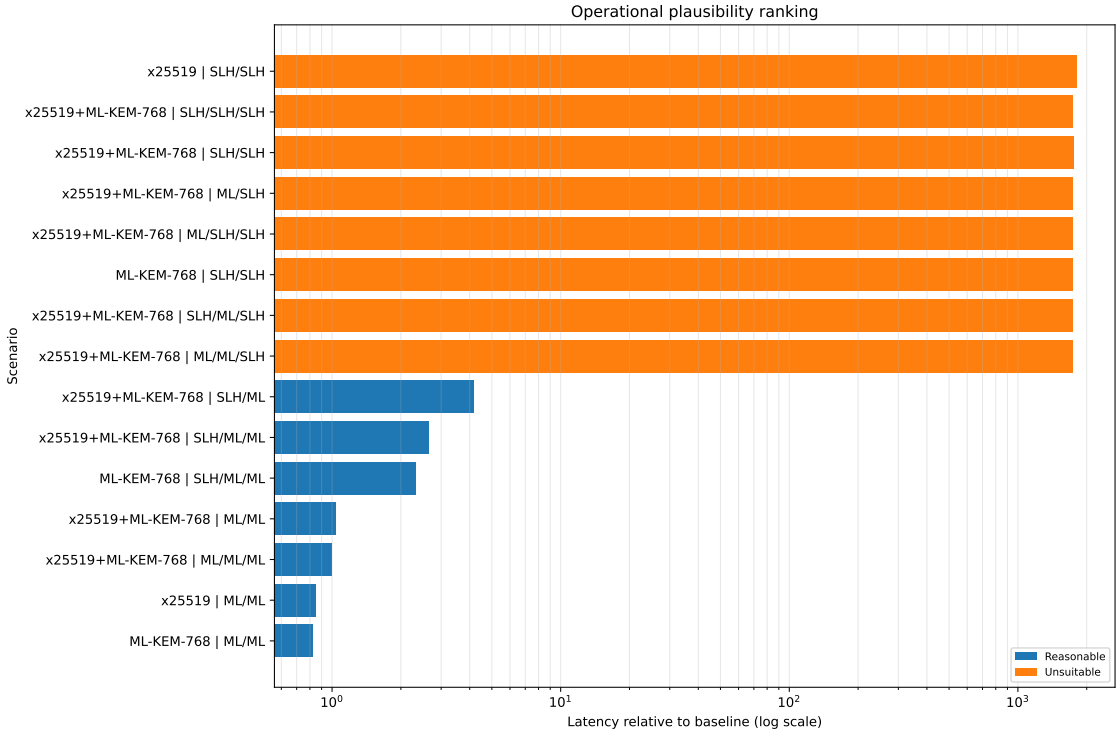


Figure 5: Operational plausibility ranking by scenario, expressed as latency relative to the fully-ML baseline.

Once SLH-DSA reaches the leaf, the finer details of the mixed hierarchy explain little of the remaining latency spread. The main explanatory break is between non-leaf-SLH and leaf-SLH configurations, not between individual mixed hierarchies.

6.3.4 Main result of Campaign B

Campaign B exposes the main strategic pattern of the study. The variable with the greatest explanatory power is not whether SLH-DSA appears somewhere in the hierarchy, but whether it appears in the leaf exposed to the live TLS handshake. Figure 5 shows the pattern compactly: the fully-ML baseline remains in the reasonable region, the `root SLH / intermediate ML / leaf ML` strategy remains penalized but plausible, and the remaining strategies move into the operationally unsuitable region once SLH-DSA reaches the leaf.

6.4 Campaign C: depth comparison

Campaign C separates topology from signature placement. This campaign shows that increasing hierarchy depth does not behave as a simple monotonic penalty. The cost of moving from depth 2 to depth 3 depends on which certificates become part of the effective chain observed during the handshake.

6.4.1 All-ML families

In the fully-ML family, additional depth does not introduce a meaningful penalty. Comparing `x25519mlkem768__ml_root__ml_leaf` with `x25519mlkem768__ml_root__ml_int__ml_leaf`, the latency ratio is effectively 1.00, with only negligible differences in observed bytes and a

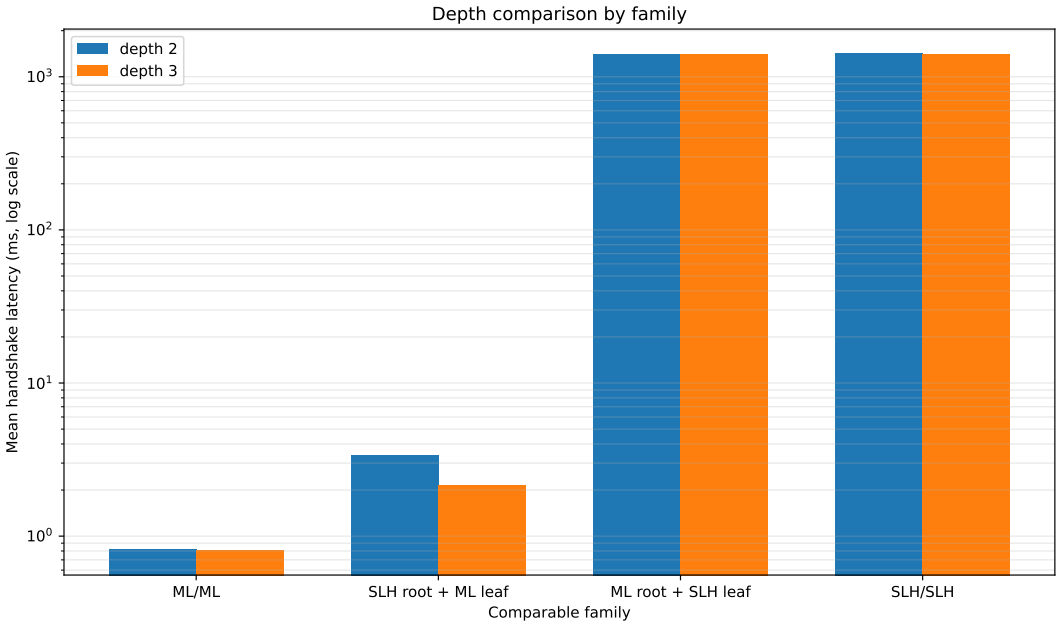


Figure 6: Depth comparison by family.

slightly smaller server task-clock in the depth-3 case. Depth alone is therefore not enough to degrade a well-behaved ML hierarchy in any meaningful way.

6.4.2 Root-SLH with ML leaf

The clearest topological effect appears when the root uses SLH-DSA while the leaf remains ML-DSA. In that family, the depth-3 variant is substantially cheaper than the depth-2 one. The latency ratio falls to approximately 0.6318, corresponding to a reduction of about 36.8% in mean latency. At the same time, `bytes_read_mean` decreases by about 11015 bytes and `chain_bytes_unique` by about 10993 bytes.

The interpretation follows from effective chain exposure. In the depth-2 variant, the observed chain corresponds to root plus leaf; in the depth-3 variant, it corresponds to intermediate plus leaf. Since the root is the heavy SLH signer in this family, removing it from the observed chain reduces visible transport and lowers overall cost materially.

6.4.3 Leaf-SLH families

The leaf-SLH families show the opposite pattern. Once SLH-DSA reaches the leaf, changes in depth still affect transport, sometimes substantially, but they have little effect on latency. In the `ML root + SLH leaf` family, the depth ratio remains effectively 1.00 even when one depth-3 variant increases observed transport by more than 11 KB. In the `SLH/SLH` family, depth may either reduce or increase observed transport, yet latency remains nearly unchanged.

This pattern indicates that depth continues to matter for transport while losing explanatory power over the main latency regime once the leaf itself becomes the dominant source of cryptographic cost.

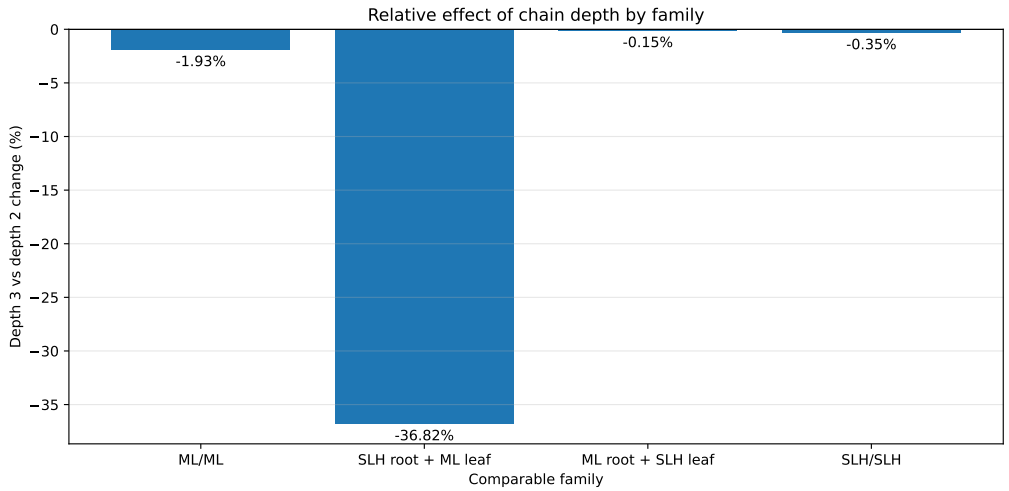


Figure 7: Relative effect of moving from depth 2 to depth 3 by comparable family, expressed as percentage change in mean latency.

6.4.4 Main result of Campaign C

Campaign C shows that hierarchy depth is not, by itself, a cost verdict. Whether additional depth penalizes, preserves, or reduces observed cost depends on how the topology changes the effective chain seen during the handshake. This matters for the interpretation of post-quantum certificate hierarchies because logical depth and practical exposure cannot be treated as interchangeable notions.

6.5 Campaign D: KEX mode exploration

Campaign D tests whether moving from classical to hybrid, or from hybrid to pure post-quantum key establishment, materially changes the placement-driven picture once comparable chains are held fixed.

In ML-leaf regimes, KEX choice modulates the cost profile. For example, moving from classical to hybrid in the simple ML-leaf case increases latency by about 1.2210×, while moving from hybrid to pure post-quantum reduces it to about 0.8220× the hybrid value. Similar but bounded effects appear in the `SLH root / ML int / ML leaf` family, where hybrid to pure post-quantum produces a latency ratio of about 0.8833×.

Once the leaf uses SLH-DSA, KEX mode contributes only local variation within the dominant heavy regime. In the `SLH root / SLH leaf` comparison, classical to hybrid yields a latency ratio of about 0.9652×, and hybrid to pure post-quantum about 0.9984×. These ratios are local perturbations inside an already severe regime.

6.5.1 Main result of Campaign D

Campaign D shows that KEX mode matters in low-cost ML-leaf scenarios and can produce moderate changes in bounded regimes. Pure post-quantum key establishment does not change the main picture. The decisive variable remains signature placement, especially the decision to place SLH-DSA in the handshake-exposed server leaf.

Table 5: Depth 2 vs. depth 3 comparisons.

pair label	depth 2 scenario_id	depth 3 scenario_id	depth 2 elapsed mean (ms)	depth 3 elapsed mean (ms)	delta elapsed mean (ms) d3 minus d2	latency ratio d3 / d2
ML/ML	x25519mlkem768__ml_root__ml_leaf	x25519mlkem768__ml_root__ml_int__ml_leaf	0.809300	0.809100	-0.000200	0.999700
SLH root + ML leaf	x25519mlkem768__slh_root__ml_leaf	x25519mlkem768__slh_root__ml_int__ml_leaf	3.376200	2.133000	-1.243100	0.631800
ML root + SLH leaf (ML intermediate)	x25519mlkem768__ml_root__slh_leaf	x25519mlkem768__ml_root__ml_int__slh_leaf	1406.283200	1402.486200	-3.797000	0.997300
ML root + SLH leaf (SLH intermediate)	x25519mlkem768__ml_root__slh_leaf	x25519mlkem768__ml_root__slh_int__slh_leaf	1406.283200	1405.802800	-0.480400	0.999700
SLH/SLH (ML intermediate)	x25519mlkem768__slh_root__slh_leaf	x25519mlkem768__slh_root__ml_int__slh_leaf	1407.714100	1403.165900	-4.548200	0.996800
SLH/SLH (SLH intermediate)	x25519mlkem768__slh_root__slh_leaf	x25519mlkem768__slh_root__slh_int__slh_leaf	1407.714100	1408.703300	0.989200	1.000700

pair label	depth 2 scenario_id	depth 3 scenario_id	delta bytes read mean d3 minus d2	delta chain bytes unique d3 minus d2	delta server task clock per run (ms) d3 minus d2	server taskclock ratio d3 / d2
ML/ML	x25519mlkem768__ml_root__ml_leaf	x25519mlkem768__ml_root__ml_int__ml_leaf	16	16	-0.040300	0.933100
SLH root + ML leaf	x25519mlkem768__slh_root__ml_leaf	x25519mlkem768__slh_root__ml_int__ml_leaf	-11015	-10993	-1.247500	0.348500
ML root + SLH leaf (ML intermediate)	x25519mlkem768__ml_root__slh_leaf	x25519mlkem768__ml_root__ml_int__slh_leaf	16	16	-3.757500	0.997300
ML root + SLH leaf (SLH intermediate)	x25519mlkem768__ml_root__slh_leaf	x25519mlkem768__ml_root__slh_int__slh_leaf	11047	11025	-1.817400	0.998700
SLH/SLH (ML intermediate)	x25519mlkem768__slh_root__slh_leaf	x25519mlkem768__slh_root__ml_int__slh_leaf	-11015	-10993	-3.238100	0.997700
SLH/SLH (SLH intermediate)	x25519mlkem768__slh_root__slh_leaf	x25519mlkem768__slh_root__slh_int__slh_leaf	16	16	-0.183900	0.999900

Table 6: KEX comparisons on comparable chains.

comparison_type	family_label	from_kex_mode	to_kex_mode	leaf_family	depth	elapsed_mean_from_ms	elapsed_mean_to_ms	latency_ratio_to_over_from
classical_vs_hybrid	ML root / ML leaf (depth 2)	classical	hybrid	ML-DSA	2	0.6885	0.8406	1.2210
classical_vs_hybrid	SLH root / SLH leaf (depth 2)	classical	hybrid	SLH-DSA	2	1464.9327	1413.9914	0.9652
hybrid_vs_pure_pqc	ML root / ML leaf (depth 2)	hybrid	pure_pqc	ML-DSA	2	0.8093	0.6652	0.8220
hybrid_vs_pure_pqc	SLH root / ML int / ML leaf (depth 3)	hybrid	pure_pqc	ML-DSA	3	2.1330	1.8842	0.8833
hybrid_vs_pure_pqc	SLH root / SLH leaf (depth 2)	hybrid	pure_pqc	SLH-DSA	2	1407.7141	1405.4559	0.9984

comparison_type	family_label	from_kex_mode	to_kex_mode	leaf_family	depth	bytes_read_from	bytes_read_to	bytes_read_ratio_to_over_from
classical_vs_hybrid	ML root / ML leaf (depth 2)	classical	hybrid	ML-DSA	2	14904	15992	1.0730
classical_vs_hybrid	SLH root / SLH leaf (depth 2)	classical	hybrid	SLH-DSA	2	49881	50969	1.0218
hybrid_vs_pure_pqc	ML root / ML leaf (depth 2)	hybrid	pure_pqc	ML-DSA	2	15992	15960	0.9980
hybrid_vs_pure_pqc	SLH root / ML int / ML leaf (depth 3)	hybrid	pure_pqc	ML-DSA	3	28947	28915	0.9989
hybrid_vs_pure_pqc	SLH root / SLH leaf (depth 2)	hybrid	pure_pqc	SLH-DSA	2	50969	50937	0.9994

comparison_type	family_label	from_kex_mode	to_kex_mode	leaf_family	depth	server_task_from_ms	server_task_to_ms	server_task_ratio_to_over_from
classical_vs_hybrid	ML root / ML leaf (depth 2)	classical	hybrid	ML-DSA	2	0.5287	0.6228	1.1780
classical_vs_hybrid	SLH root / SLH leaf (depth 2)	classical	hybrid	SLH-DSA	2	1462.2813	1411.3448	0.9652
hybrid_vs_pure_pqc	ML root / ML leaf (depth 2)	hybrid	pure_pqc	ML-DSA	2	0.6022	0.5152	0.8555
hybrid_vs_pure_pqc	SLH root / ML int / ML leaf (depth 3)	hybrid	pure_pqc	ML-DSA	3	0.6672	0.5273	0.7903
hybrid_vs_pure_pqc	SLH root / SLH leaf (depth 2)	hybrid	pure_pqc	SLH-DSA	2	1405.0874	1402.8960	0.9984

7 Cross-Cutting Cryptographic Interpretation

The campaign-based results establish the empirical structure of the study. Reorganizing the evidence around four cross-cutting questions clarifies that structure: whether signature placement explains behavior better than signature-family presence alone, whether transport-related overhead is sufficient to explain the heavy regime, whether effective chain exposure must be analyzed as part of the authentication path, and whether the leaf-SLH cases form a distinct operational regime or the upper tail of a continuous cost spectrum. Under this organization, the results support an interpretation of post-quantum TLS authentication as a hierarchy-sensitive cryptographic design problem.

7.1 Signature placement versus mere signature presence

The central empirical finding is that signature placement explains the observed authentication regimes better than signature-family presence alone. When scenarios are grouped by placement class rather than by campaign, the structure of the data becomes more clearly separated.

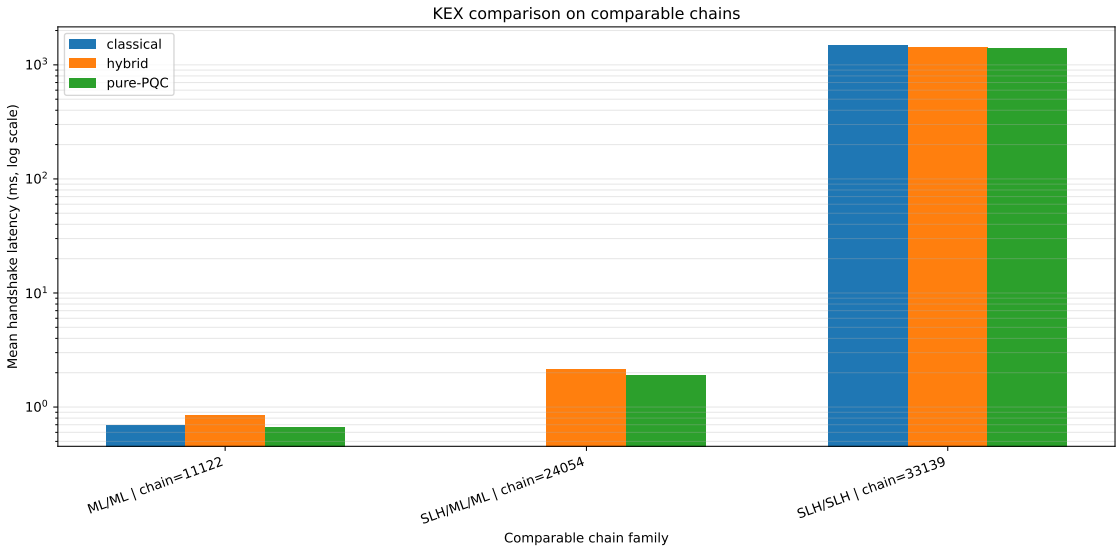


Figure 8: KEX comparison on comparable chains.

Table 7: Signature placement summary.

placement_class	n_scenarios	mean_elapsed_ms	median_elapsed_ms	min_elapsed_ms	max_elapsed_ms	mean_latency_vs_baseline
all_ml	5	0.763	0.809	0.665	0.841	0.942
root_slh_leaf_not_slh	3	2.464	2.133	1.884	3.376	3.046
intermediate_slh_any	2	1407.253	1407.253	1405.803	1408.703	1739.386
leaf_slh	9	1413.171	1406.283	1402.486	1464.933	1746.701

placement_class	median_latency_vs_baseline	mean_bytes_vs_baseline	mean_server_cpu_vs_baseline	mean_server_over_elapsed	mean_client_over_elapsed
all_ml	1.000	0.985	1.008	0.744	0.742
root_slh_leaf_not_slh	2.636	2.037	1.844	0.387	0.803
intermediate_slh_any	1739.386	2.781	2498.686	0.998	0.003
leaf_slh	1738.188	2.678	2510.849	0.998	0.002

Figure 9 and Table 7 organize the scenario space into four categories: all-ML, root-SLH with non-SLH leaf, intermediate-SLH-any, and leaf-SLH. The all-ML class defines the low-cost reference regime of the study. Its mean latency is approximately 0.7625 ms, with mean server-side compute essentially aligned with the baseline. These scenarios form a tight and stable low-cost cluster.

The `root-SLH / leaf-not-SLH` class is clearly more expensive than all-ML, but it remains within a bounded interactive regime. Its mean latency rises to approximately 2.4645 ms, corresponding to a mean multiplier of about 3.0461× relative to baseline, while server-side compute rises by about 1.8444×. The presence of SLH-DSA somewhere in the hierarchy therefore does not by itself force the handshake into the catastrophic region.

By contrast, the `leaf-SLH` class defines the heavy regime of the study. Its mean latency reaches approximately 1413.1706 ms, with a mean latency multiplier of approximately 1746.7007× relative to baseline and a mean server-side compute multiplier of approximately 2510.8488×. This contrast is too large to read as a continuation of the upper-layer SLH penalty. Moving from all-ML to upper-layer SLH produces a bounded increase, whereas moving from upper-layer SLH to leaf-SLH produces a discontinuity.

This is the clearest expression of the paper's main thesis. The decisive variable is not whether SLH-DSA appears somewhere in the certification path, but whether it reaches the handshake-

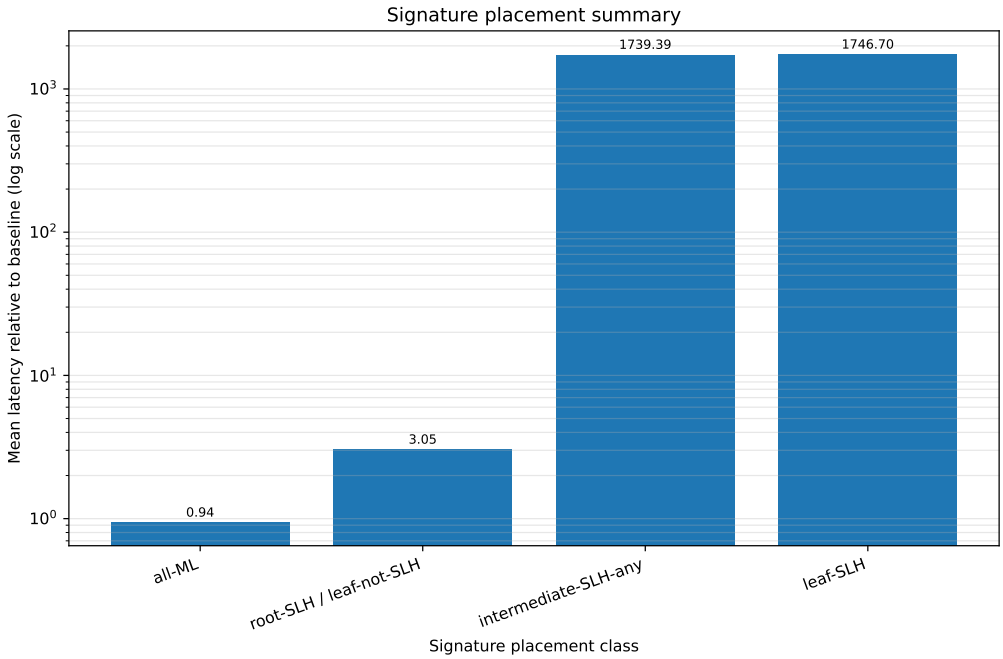


Figure 9: Signature placement summary.

Table 8: Transport-versus-latency correlations.

subset	n_scenarios	metric	pearson_r	spearman_rho
all_scenarios	17	bytes_read_mean	0.7493	0.8503
all_scenarios	17	chain_bytes_unique	0.3943	0.5227
non_leaf_slh	8	bytes_read_mean	0.9937	0.8982
non_leaf_slh	8	chain_bytes_unique	0.9933	0.8045
leaf_slh_only	9	bytes_read_mean	0.3518	0.5523
leaf_slh_only	9	chain_bytes_unique	0.3826	0.5919

exposed leaf. Hierarchy position therefore has greater explanatory force than signature-family presence taken in the abstract.

7.2 Transport-related overhead versus cryptographic cost

Post-quantum TLS evaluation can easily reduce certificate overhead to wire cost. The data support a more limited interpretation. Transport-related overhead is relevant and, outside the heavy regime, often highly informative. It is not sufficient to explain the catastrophic behavior associated with leaf-SLH scenarios.

Across the full dataset, latency exhibits a moderate positive relationship with `bytes_read_mean` and a weaker one with `chain_bytes_unique`. This could suggest that the heavy scenarios are the consequence of larger transmitted objects. Once leaf-SLH scenarios are excluded, however, latency in the non-catastrophic region tracks both `bytes_read_mean` and `chain_bytes_unique` very closely. Transport explains much of the variation while the system remains in a bounded regime.

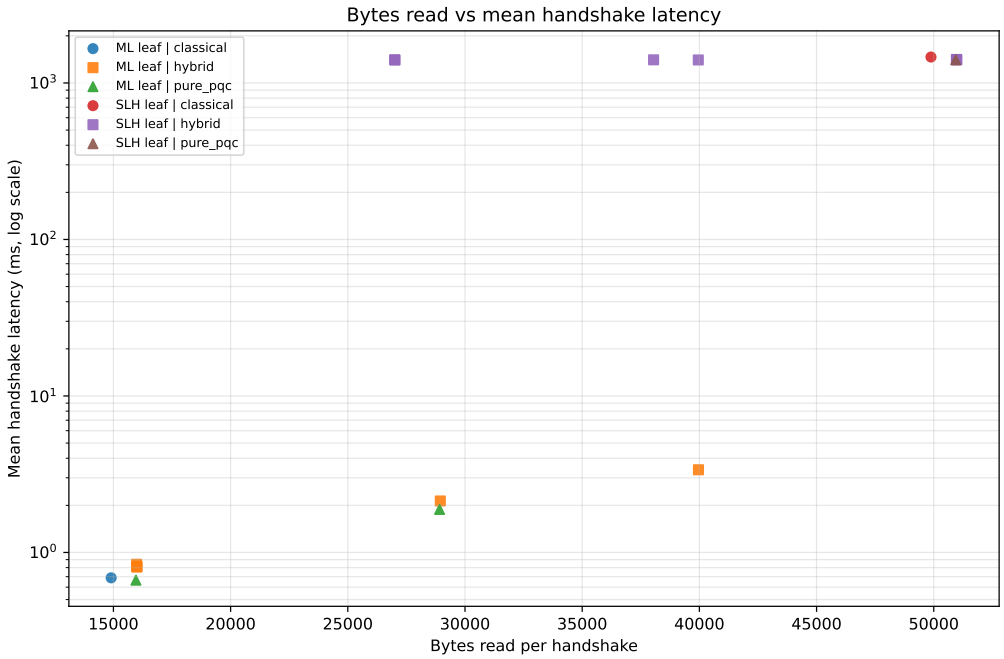


Figure 10: Bytes read versus mean handshake latency.

The picture changes within the leaf-SLH cases. In that region, the relation between latency and transport weakens sharply. Very large latency persists despite only moderate variation in transferred bytes and observed chain size. Figures 10 and 11 show this geometrically: outside leaf-SLH, the points follow a relatively coherent growth pattern; inside leaf-SLH, they collapse into a heavy cloud whose internal latency variation is only weakly explained by transport.

The counterexamples in Table 9 are especially informative. A scenario may transmit more bytes and still remain hundreds of times faster than another scenario that transmits fewer bytes, provided the former keeps SLH-DSA out of the leaf while the latter places it directly in the interactive certificate. That pattern cannot be reduced to wire expansion alone.

The interpretation is layered. Transport overhead is real and remains highly relevant in bounded regimes. Once SLH-DSA reaches the leaf, however, the dominant explanatory variable changes. At that point, the principal cost source is concentrated cryptographic work during live authentication.

7.3 Effective chain exposure as a first-class variable

The transport findings become fully intelligible only when effective chain exposure is treated as a first-class analytical variable. The logical PKI hierarchy is not the whole story. What matters for the handshake is the chain that is actually exposed and processed during protocol execution.

Campaign C provides the clearest evidence for this point. In the `root-SLH / leaf-ML` family, moving from depth 2 to depth 3 reduces latency substantially instead of increasing it. In the depth 2 case, the observed chain corresponds to root plus leaf; in the depth 3 case, it corresponds to intermediate plus leaf. Since the root is the heavy SLH signer in that family, removing it from the effective chain reduces visible transport and lowers overall cost materially.

Effective chain exposure is part of the cryptographic-protocol phenomenon under study. It determines which certificates are transmitted, which signature objects enter the authenticated path, and which validation burden becomes visible during the handshake.

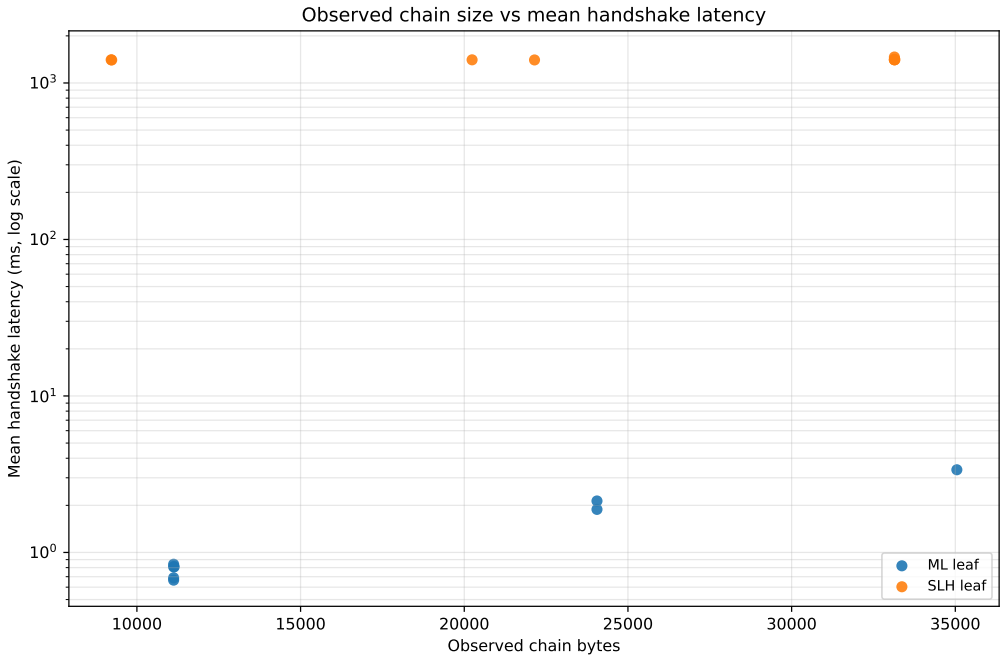


Figure 11: Observed chain size versus mean handshake latency.

The point is especially important in post-quantum settings because certificate families may differ sharply in footprint and processing cost. Under those conditions, the gap between declared hierarchy and effective exposure can become operationally decisive. A study that models only logical hierarchy depth, while ignoring which certificates appear in the live chain, risks missing one of the mechanisms through which authentication cost is shaped.

For that reason, this paper treats effective chain exposure as one of the main explanatory variables. The distinction between root-plus-leaf and intermediate-plus-leaf is one of the mechanisms through which hierarchy design changes the cryptographic surface of the TLS handshake.

7.4 Emergence of a distinct leaf-SLH regime

The evidence presented so far supports a stronger claim than the statement that some scenarios are slower than others. It supports the conclusion that the leaf-SLH cases form a distinct operational regime.

This regime is characterized by four simultaneous properties. First, it exhibits a stable latency plateau around 1.4 seconds rather than a broad gradient of increasingly bad cases. Second, it is tightly linked to signature placement: once SLH-DSA reaches the leaf, the heavy regime appears consistently across campaigns, topology variants, and KEX modes. Third, its relation to transport weakens sharply, as shown by the transport correlations and the wire-size counterexamples. Fourth, it is comparatively insensitive to changes in key-establishment mode, as Campaign D shows.

Taken together, these properties indicate that leaf-SLH is not simply the expensive end of an otherwise continuous design space. It is a qualitatively different authentication regime with its own internal regularity. The scenarios in that class vary in transmitted bytes, chain composition, and hierarchy structure, but they remain confined to a common heavy band whose defining feature is the placement of SLH-DSA in the handshake-exposed leaf.

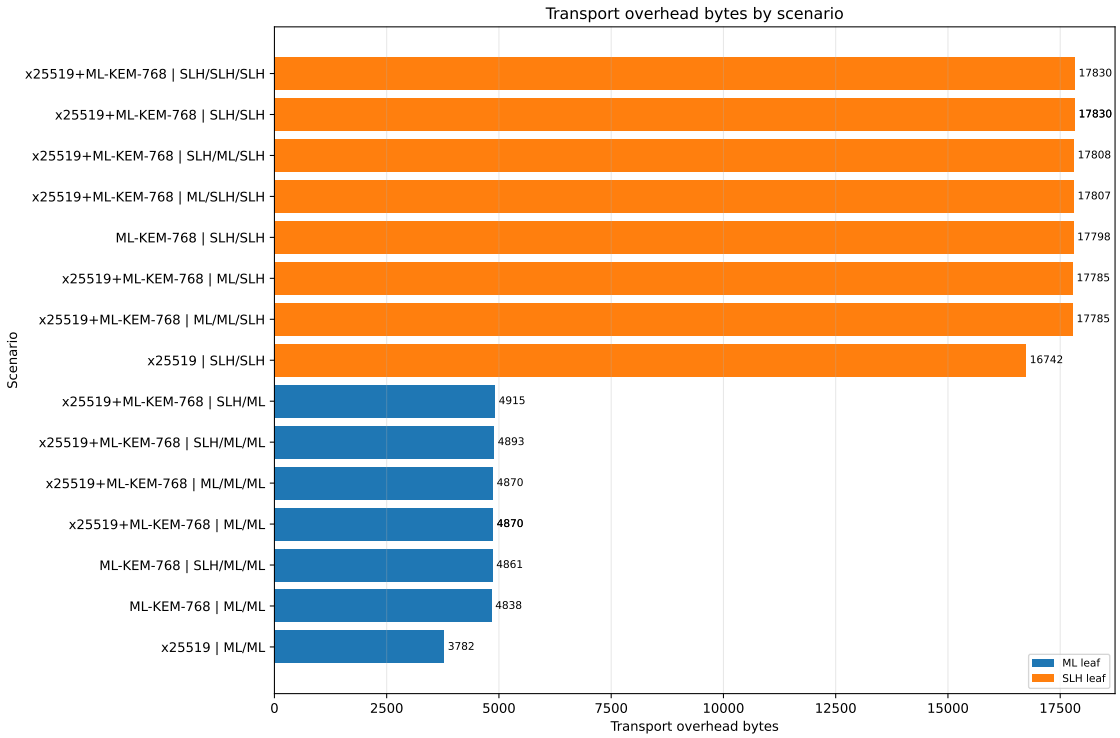


Figure 12: Transport overhead bytes by scenario.

The paper therefore treats leaf-SLH as a regime concept rather than as a single bad design point. The experimental campaigns reveal a recurring structure in which certificate placement drives the handshake into a state that is only weakly explained by transport, highly stable across scenario variants, and operationally distinct from both the all-ML baseline and the bounded upper-layer-SLH cases.

The next section examines where the cost of this regime resides. If leaf-SLH defines a distinct state of TLS authentication, the next causal question is whether that state is balanced across both peers, validation-skewed, or overwhelmingly concentrated on the server side.

8 Client/Server Cost Decomposition

The previous sections established that leaf-SLH scenarios form a distinct heavy regime and that transport-related overhead alone does not explain their cost. This section examines where active cryptographic work is incurred during the handshake by decomposing the authentication path into client- and server-side measurements.

8.1 Limits of client-only observations

Earlier stages of the TLS benchmarking work suggested that client-side activity could not account for the second-scale latency observed in the heaviest scenarios. Client-only observations were nevertheless incomplete: they could indicate that the client did not consume enough active CPU time to justify the total elapsed duration, but they could not identify where the remaining time was spent.

Table 9: Counterexamples to wire-size-only explanations.

rank	transport metric	scenario more bytes lower latency	scenario less bytes higher latency	more bytes value	less bytes value	bytes diff	lower latency (ms)	higher latency (ms)	latency ratio higher / lower
1	bytes_read_mean	x25519mlkem768__slh_root__ml_leaf	x25519mlkem768__ml_root__slh_leaf	39962.0000	26999.0000	12963.0000	3.3762	1406.2832	416.5316
2	bytes_read_mean	x25519mlkem768__slh_root__ml_leaf	x25519mlkem768__ml_root__ml_int__slh_leaf	39962.0000	27015.0000	12947.0000	3.3762	1402.4862	415.4070
3	bytes_read_mean	mlkem768__slh_root__ml_int__ml_leaf	x25519mlkem768__ml_root__slh_leaf	28915.0000	26999.0000	1916.0000	1.8842	1406.2832	746.3713
4	bytes_read_mean	x25519mlkem768__slh_root__ml_int__ml_leaf	x25519mlkem768__ml_root__slh_leaf	28947.0000	26999.0000	1948.0000	2.1330	1406.2832	659.2901
5	bytes_read_mean	mlkem768__slh_root__ml_int__ml_leaf	x25519mlkem768__ml_root__ml_int__slh_leaf	28915.0000	27015.0000	1900.0000	1.8842	1402.4862	744.3561

rank	transport metric	scenario more bytes lower latency	scenario less bytes higher latency	more bytes value	less bytes value	bytes diff	lower latency (ms)	higher latency (ms)	latency ratio higher / lower
1	chain_bytes_unique	x25519mlkem768__slh_root__ml_leaf	x25519mlkem768__ml_root__slh_leaf	35047.0000	9214.0000	25833.0000	3.3762	1406.2832	416.5316
2	chain_bytes_unique	x25519mlkem768__slh_root__ml_leaf	x25519mlkem768__ml_root__ml_int__slh_leaf	35047.0000	9230.0000	25817.0000	3.3762	1402.4862	415.4070
3	chain_bytes_unique	mlkem768__slh_root__ml_int__ml_leaf	x25519mlkem768__ml_root__slh_leaf	24054.0000	9214.0000	14840.0000	1.8842	1406.2832	746.3713
4	chain_bytes_unique	mlkem768__slh_root__ml_int__ml_leaf	x25519mlkem768__ml_root__ml_int__slh_leaf	24054.0000	9230.0000	14824.0000	1.8842	1402.4862	744.3561
5	chain_bytes_unique	x25519mlkem768__slh_root__ml_int__ml_leaf	x25519mlkem768__ml_root__slh_leaf	24054.0000	9214.0000	14840.0000	2.1330	1406.2832	659.2901

A slow handshake may result from heavier client-side validation, heavier server-side authentication work, roughly symmetric stress on both peers, or waiting behavior dominated by one peer while the other remains active. Without server-side measurements, those possibilities remain partially entangled. The inclusion of server-side performance counters in the final dataset closes that gap and allows the heavy regime to be interpreted in workload terms rather than only in latency terms.

The decomposition distinguishes among three possible structures for the leaf-SLH regime: balanced work across client and server, a shift toward client-side validation, or overwhelming concentration of active work on the server side.

8.2 Balanced regime in all-ML scenarios

The all-ML class remains the reference regime in both latency and workload structure. Across those scenarios, mean client task-clock is approximately 0.5647 ms, while mean server task-clock is approximately 0.5662 ms. Normalized by elapsed time, the client contributes about 0.7423 of elapsed time in active work and the server about 0.7444. The mean server/client task-clock ratio is approximately 1.0043, and the mean server/client instruction ratio is approximately 1.0841.

These values describe a regime that is low-cost and structurally balanced. Both ends of the handshake perform comparable amounts of active work, and neither dominates the other. Figure 14 shows this directly: in the low-cost all-ML region, client and server bars remain of similar magnitude. Figure 15 shows the same structure geometrically, with all-ML scenarios clustering close to the parity line. Figure 17 reinforces the point through ratios close to one.

This balanced structure provides the control case against which the remaining regimes can be interpreted. Without it, the heavy scenarios might be read simply as more expensive instances of the same general phenomenon. The decomposition shows that they differ qualitatively in where active work resides.

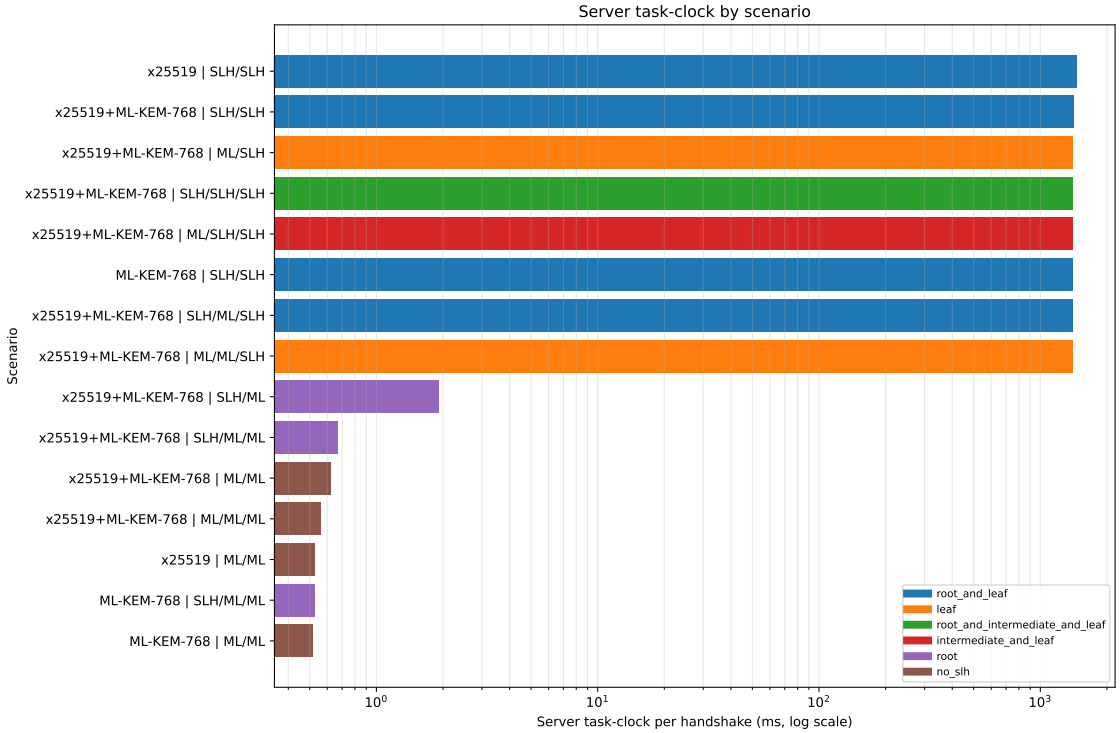


Figure 13: Server task-clock by scenario.

8.3 Validation-skewed regime in upper-layer SLH scenarios

The next distinct regime is formed by scenarios in which SLH-DSA appears in upper trust layers while the interactive leaf remains ML-DSA. These cases incur a visible performance penalty while remaining operationally plausible, and their decomposition shows that the penalty is not distributed symmetrically.

For the `root-SLH / leaf-ML` class, mean client task-clock rises to approximately 1.8708 ms, whereas mean server task-clock rises only to about 1.0364 ms. When normalized by elapsed time, the client contributes approximately 0.8032 of elapsed time in active work, while the server contributes only about 0.3866. The mean server/client task-clock ratio falls to approximately 0.5477, and the mean instruction ratio to about 0.4360.

This is a validation-skewed regime rather than a server-collapse pattern. The server does more work than in the all-ML baseline, but the relative burden shifts more strongly toward the client. The most natural interpretation is that placing SLH-DSA above the leaf increases the validation burden visible on the client side without pushing the interactive server-authentication step into the catastrophic regime.

The contrast with the full strategy matrix makes this distinction clear. The `x25519mlkem768__slh_root__ml_int__ml_leaf` strategy is materially slower than the fully-ML baseline, but it does not resemble the leaf-SLH cases in workload structure. Geometrically, these scenarios move away from the parity cluster without entering the vertically separated server-dominated cloud shown in Figure 15. That difference indicates that upper-layer SLH and leaf-SLH are not weaker and stronger versions of the same phenomenon.

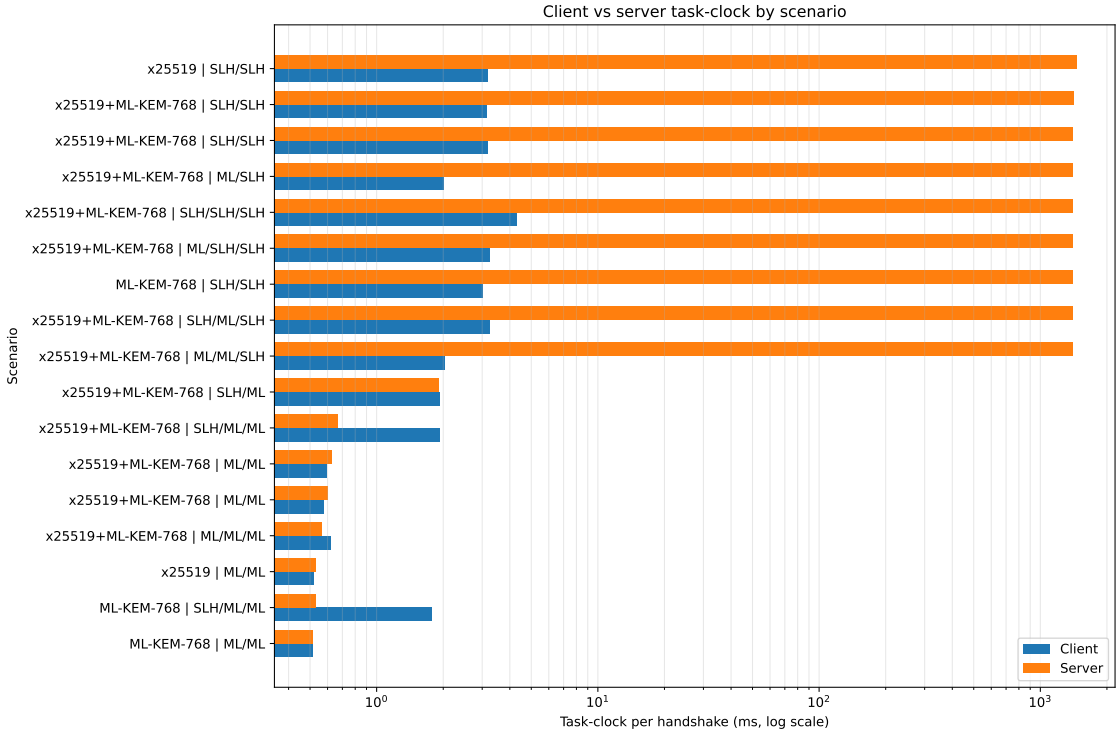


Figure 14: Client versus server task-clock by scenario.

8.4 Overwhelmingly server-bound regime in leaf-SLH scenarios

The decomposition changes sharply once SLH-DSA reaches the leaf. In that class, mean client task-clock is only about 3.0402 ms, while mean server task-clock rises to approximately 1410.8409 ms. The normalized ratios are even more informative: the client contributes only about 0.0022 of elapsed time in active work, whereas the server contributes about 0.9984. The mean server/client task-clock ratio is approximately 487.95, and the mean server/client instruction ratio is approximately 648.91.

At that point the handshake becomes overwhelmingly server-bound. Figure 13 shows that server-side task-clock grows sharply only in the leaf-SLH cases. Figure 15 shows the corresponding geometric separation: leaf-SLH scenarios form a distinct cloud far above the client/server parity line rather than a gradual extension of the low-cost region. Figure 16 makes the same point directly. In those scenarios, server task-clock almost coincides with end-to-end elapsed time. In practical terms, the total handshake duration is almost entirely accounted for by server-side active compute, while the client is left largely in a waiting role.

Figure 17 summarizes the same asymmetry. Ratios in leaf-SLH scenarios jump from near-unity or sub-unity values into the hundreds. Some of the most extreme cases show server/client task-clock ratios above 680 and instruction ratios above 1000. The handshake is dominated by server-side work to an extent that marks a clear regime change.

Among the most server-dominated scenarios are `x25519mlkem768__slh_root__ml_int__slh_leaf`, `x25519mlkem768__ml_root__ml_int__slh_leaf`, and `x25519mlkem768__ml_root__slh_leaf`, all of which exhibit `server_taskclock_over_elapsed` ratios around 0.999 together with very large server/client asymmetries. The decomposition therefore sharpens the central claim of the

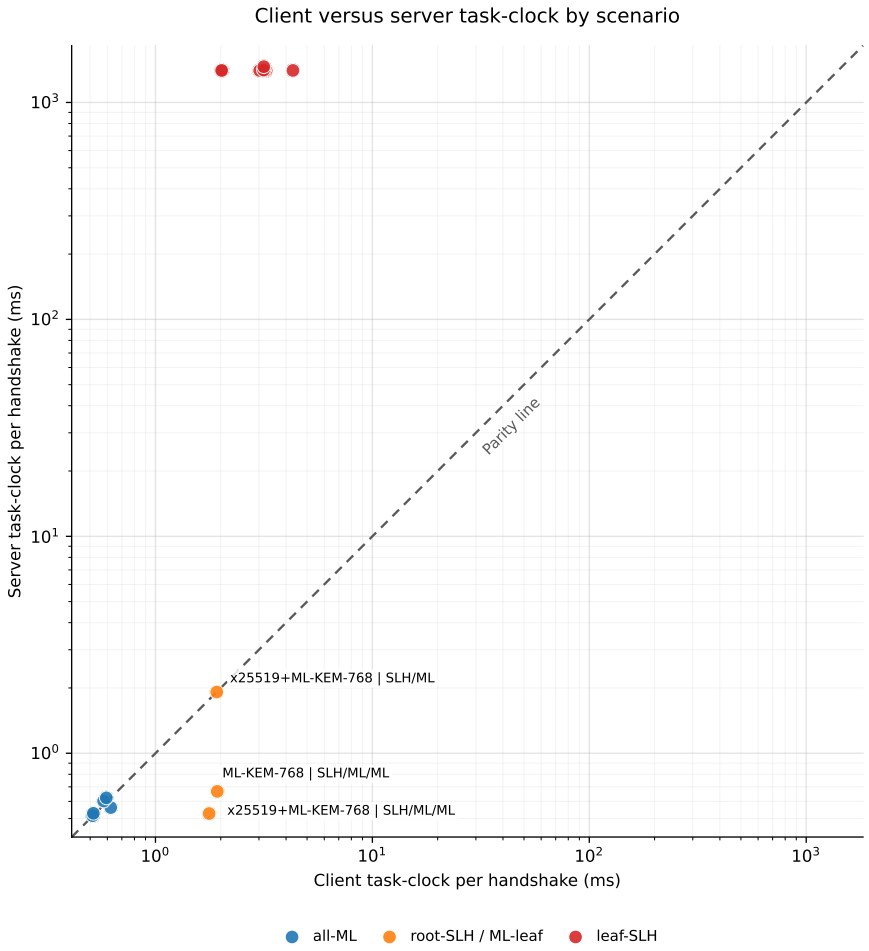


Figure 15: Client versus server task-clock by scenario on log-log axes. The diagonal marks parity between client-side and server-side active work; the clustered separation reveals distinct authentication regimes.

paper into a causal statement. The degradation observed in leaf-SLH scenarios is neither primarily a transport phenomenon nor primarily a client-validation phenomenon. It is overwhelmingly a server-side cryptographic compute phenomenon concentrated in the live authentication path.

8.5 Supporting microarchitectural signals

The dataset also includes lower-level counters such as IPC, cache-miss rate, and branch-miss rate. The present study is not a function-tracing or microarchitectural reverse-engineering exercise, and these counters are used only as supporting evidence for regime differentiation rather than as proof of a unique internal mechanism.

With that caution in place, the counters still suggest that the heavy leaf-SLH regime is not simply more of the same workload. In the low-cost ML scenarios, the server tends to operate in a comparatively modest regime, with IPC values around 2.1–2.35 and cache-miss rates in the low single-digit range. By contrast, the heavy leaf-SLH scenarios exhibit a stable and sharply separated profile, with server IPC around 4.54–4.55, cache-miss rates roughly in the 33–40% range, and branch-miss rates around 0.48–0.51%.

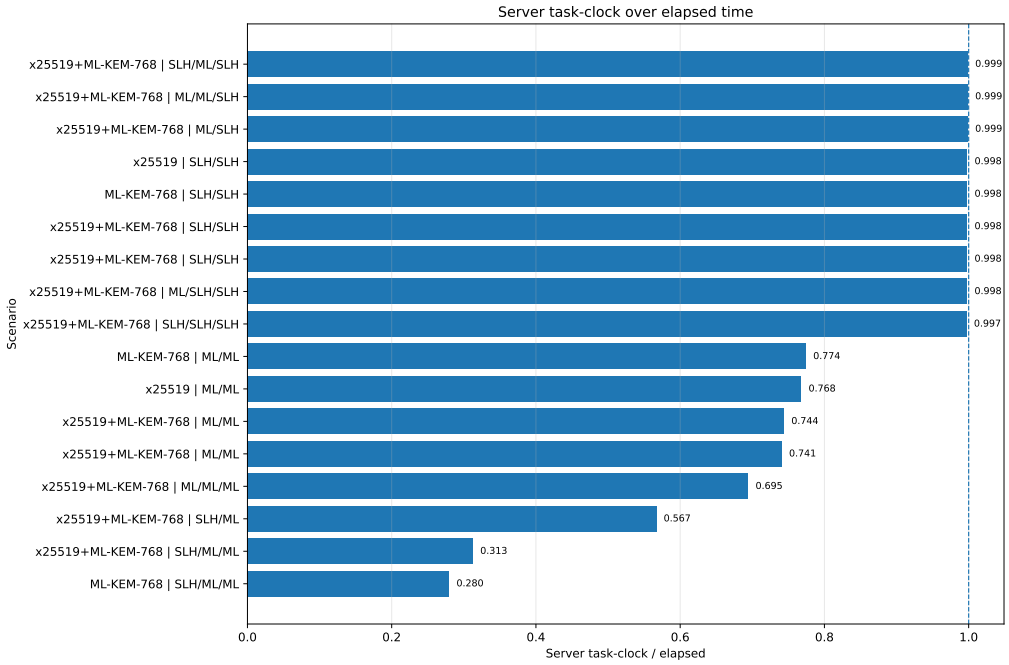


Figure 16: Server task-clock over elapsed time.

The paper does not identify a single internal OpenSSL or provider function as the unique source of cost, nor does it reconstruct the complete internal execution path. The defensible point is that the leaf-SLH cases exhibit a stable server-side execution profile that is sharply distinct from the low-cost ML regime.

Taken together, the decomposition results complete the causal arc of the paper. The previous section showed that leaf-SLH defines a distinct authentication regime that transport alone does not explain. The present section shows why: once SLH-DSA reaches the server leaf, the handshake becomes almost entirely a server-side compute event. That finding provides the bridge from hierarchy design to deployment consequence and prepares the next step of the argument, namely the translation of authentication cost into capacity loss and operational viability.

9 Operational Implications for PQ TLS Deployment

The previous sections showed that the dominant heavy regime is overwhelmingly server-side and concentrated in leaf-SLH scenarios. The operational question is how second-scale server-authentication cost should be translated into capacity loss, infrastructure scaling, and deployment plausibility for a service operator.

9.1 From server-side handshake cost to service capacity

For an interactive TLS service, handshake cost is both a latency issue and a capacity issue. Every additional unit of server-side work per handshake reduces the number of sessions a node can sustain over time, lowers effective throughput per core, and increases the infrastructure required to maintain a fixed service rate. In high-volume settings, these effects accumulate quickly. A per-handshake cryptographic penalty therefore becomes a constraint on concurrency, scaling, and operational headroom.

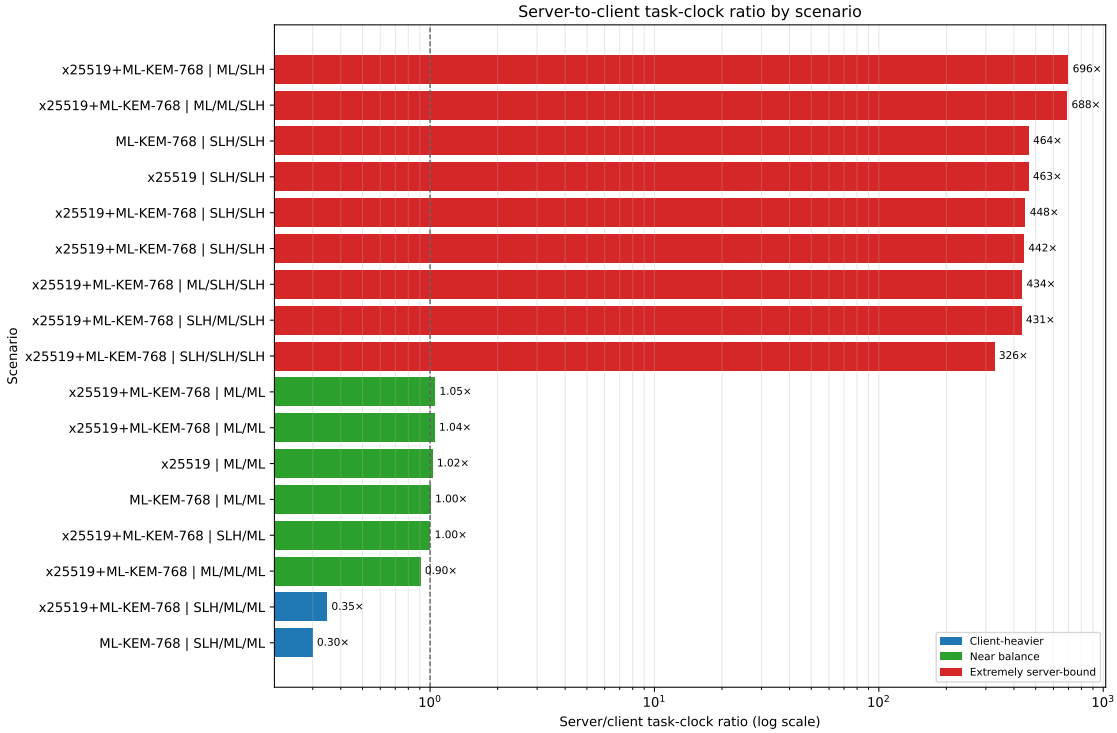


Figure 17: Server-to-client task-clock ratio by scenario.

The interpretation must therefore go beyond raw latency. Once the client/server decomposition showed that the heavy regime is overwhelmingly server-side, the relevant unit of interpretation became server CPU time per handshake. The translation to service capacity then follows directly: if one authentication strategy consumes orders of magnitude more server compute per handshake than another, throughput per node falls proportionally, and sustaining the same rate of authenticated connections requires proportionally more capacity.

Under that view, the heavy leaf-SLH regime is an authentication design that consumes server compute in a way that directly constrains deployable scale. For an operator, the relevant question is whether the handshake remains viable at the service layer.

9.2 Capacity loss under leaf-SLH

The most direct operational translation of the server-side measurements is handshake capacity. Table 11 reports derived capacity metrics such as handshakes per core-second, handshakes per vCPU-hour, retained capacity relative to baseline, and the infrastructure multiplier required to preserve baseline throughput.

The hybrid depth-3 fully-ML baseline, `x25519mlkem768__ml_root__ml_int__ml_leaf`, supports approximately 1779.68 handshakes per core-second, or about 6,406,856.76 handshakes per vCPU-hour. This is the reference point against which the remaining strategies are interpreted.

The all-ML family remains essentially aligned with that baseline. Its mean retained capacity is approximately 0.9977× baseline, and its mean infrastructure multiplier is approximately 1.0076×. In practical terms, the all-ML region is capacity-stable.

Table 10: Performance decomposition summary.

scenario_id	elapsed_mean_ms	client_task_clock_per_run_ms	server_task_clock_per_run_ms	client_taskclock_over_elapsed
mlkem768__ml_root__ml_leaf	0.665200	0.513600	0.515200	0.772100
mlkem768__slh_root__ml_int__ml_leaf	1.884200	1.767500	0.527300	0.938100
mlkem768__slh_root__slh_leaf	1405.455900	3.024100	1402.896000	0.002200
x25519__leaf_mldsa65	0.688500	0.517300	0.528700	0.751400
x25519__leaf_slhdsashake192s	1464.932700	3.159200	1462.281300	0.002200
x25519mlkem768__leaf_mldsa65	0.840600	0.593700	0.622800	0.706200
x25519mlkem768__leaf_slhdsashake192s	1413.991400	3.148500	1411.344800	0.002200
x25519mlkem768__ml_root__ml_int__ml_leaf	0.809100	0.622200	0.561900	0.769000
x25519mlkem768__ml_root__ml_int__slh_leaf	1402.486200	2.036700	1401.169400	0.001500
x25519mlkem768__ml_root__ml_leaf	0.809300	0.576800	0.602200	0.712700
x25519mlkem768__ml_root__slh_int__slh_leaf	1405.802800	3.235800	1403.109500	0.002300
x25519mlkem768__ml_root__slh_leaf	1406.283200	2.018000	1404.926900	0.001400
x25519mlkem768__slh_root__ml_int__ml_leaf	2.133000	1.927300	0.667200	0.903600
x25519mlkem768__slh_root__ml_int__slh_leaf	1403.165900	3.254300	1401.849300	0.002300
x25519mlkem768__slh_root__ml_leaf	3.376200	1.917600	1.914600	0.568000
x25519mlkem768__slh_root__slh_int__slh_leaf	1408.703300	4.303500	1404.903400	0.003100
x25519mlkem768__slh_root__slh_leaf	1407.714100	3.182100	1405.087400	0.002300

scenario_id	server_taskclock_over_elapsed	server_client_taskclock_ratio	qualitative_perf_regime
mlkem768__ml_root__ml_leaf	0.774500	1.003000	balanced
mlkem768__slh_root__ml_int__ml_leaf	0.279800	0.298300	client_skewed
mlkem768__slh_root__slh_leaf	0.998200	463.910400	overwhelmingly_server_bound
x25519__leaf_mldsa65	0.768000	1.022100	balanced
x25519__leaf_slhdsashake192s	0.998200	462.869300	overwhelmingly_server_bound
x25519mlkem768__leaf_mldsa65	0.741000	1.049200	balanced
x25519mlkem768__leaf_slhdsashake192s	0.998100	448.264200	overwhelmingly_server_bound
x25519mlkem768__ml_root__ml_int__ml_leaf	0.694500	0.903100	balanced
x25519mlkem768__ml_root__ml_int__slh_leaf	0.999100	687.949400	overwhelmingly_server_bound
x25519mlkem768__ml_root__ml_leaf	0.744100	1.044100	balanced
x25519mlkem768__ml_root__slh_int__slh_leaf	0.998100	433.616100	overwhelmingly_server_bound
x25519mlkem768__ml_root__slh_leaf	0.999000	696.186200	overwhelmingly_server_bound
x25519mlkem768__slh_root__ml_int__ml_leaf	0.312800	0.346200	client_skewed
x25519mlkem768__slh_root__ml_int__slh_leaf	0.999100	430.763900	overwhelmingly_server_bound
x25519mlkem768__slh_root__ml_leaf	0.567100	0.998500	balanced
x25519mlkem768__slh_root__slh_int__slh_leaf	0.997300	326.456000	overwhelmingly_server_bound
x25519mlkem768__slh_root__slh_leaf	0.998100	441.559800	overwhelmingly_server_bound

The `root-SLH / leaf-ML` family introduces a real but bounded degradation. Its mean retained capacity is approximately 0.7338× baseline, implying a mean infrastructure multiplier of approximately 1.8444×, with a much more moderate median multiplier of approximately 1.1874×. This is not negligible, but it remains within a range that could be tolerated in some deployment contexts, especially where a heavier upper trust layer is judged worthwhile.

The `leaf-SLH` class behaves very differently. Its mean retained capacity falls to approximately 0.0004× baseline, and its mean infrastructure multiplier reaches approximately 2510.85×, with a median around 2500.28×. Figure 18 shows the scale of this discontinuity. The leaf-SLH scenarios stand several orders of magnitude beyond baseline.

At that point, the architectural distinction becomes operationally unavoidable. A configuration that requires on the order of 2500× more infrastructure to preserve baseline throughput cannot reasonably be described only as less efficient. It belongs to a different deployment class altogether. The language of bounded penalty no longer applies; the observed effect is a collapse of handshake throughput as a practical front-end primitive.

9.3 Cost-per-million-handshakes interpretation

Capacity loss is already an infrastructure result, but cost per million handshakes provides a second operational translation. This metric is portable across traffic scales, easy to compare, and tied directly to per-handshake server burden.

Table 12 and Figure 19 summarize this view. Under the illustrative price model used in the analysis, the fully-ML baseline costs approximately 0.006243 per million handshakes. The `root-SLH /`

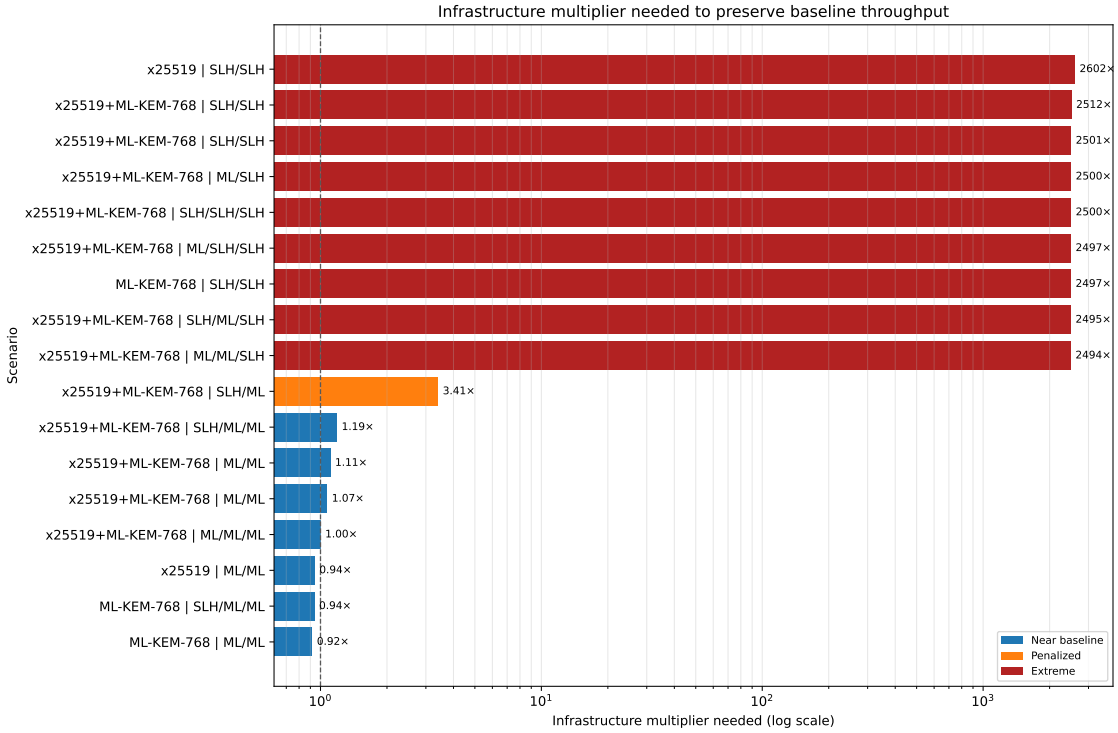


Figure 18: Infrastructure multiplier needed to preserve baseline throughput.

`ml_int / ml_leaf` strategy rises only modestly, to approximately 0.007413 per million, for an extra cost of about 0.001170 and a cost multiplier of about 1.187×.

The leaf-SLH cases expand economically in direct parallel with their compute behavior. Representative heavy scenarios such as `x25519mlkem768__slh_root__slh_int__slh_leaf` and `x25519mlkem768__ml_root__slh_leaf` land around 15.61 cost units per million handshakes, yielding extra costs above 15.60 relative to baseline and cost multipliers near 2500×. The most extreme observed case, `x25519__leaf_slhdsashake192s`, reaches approximately 16.247570 per million handshakes, or about 2602.396× the baseline.

This metric compresses the operational meaning of measured server-side cost into a quantity that remains intelligible across contexts. In that representation, upper-layer SLH appears as a bounded premium. Leaf-SLH appears as a structural departure from the baseline rather than as an expensive ordinary variant.

9.4 Service-class translation

Cost per million handshakes is portable, but operators often reason in deployment classes rather than in normalized per-million metrics. For that reason, the analysis also expresses the economic model through three illustrative service classes: `small_internal`, `medium_api`, and `high_volume_frontend`.

Table 13 and Figure 20 present that translation. Under the illustrative assumptions of the model, the median extra monthly cost of the `root-SLH / leaf-ML` class remains small: around 0.0035 for a small internal service, around 0.3509 for a medium API, and around 3.5092 for a high-volume

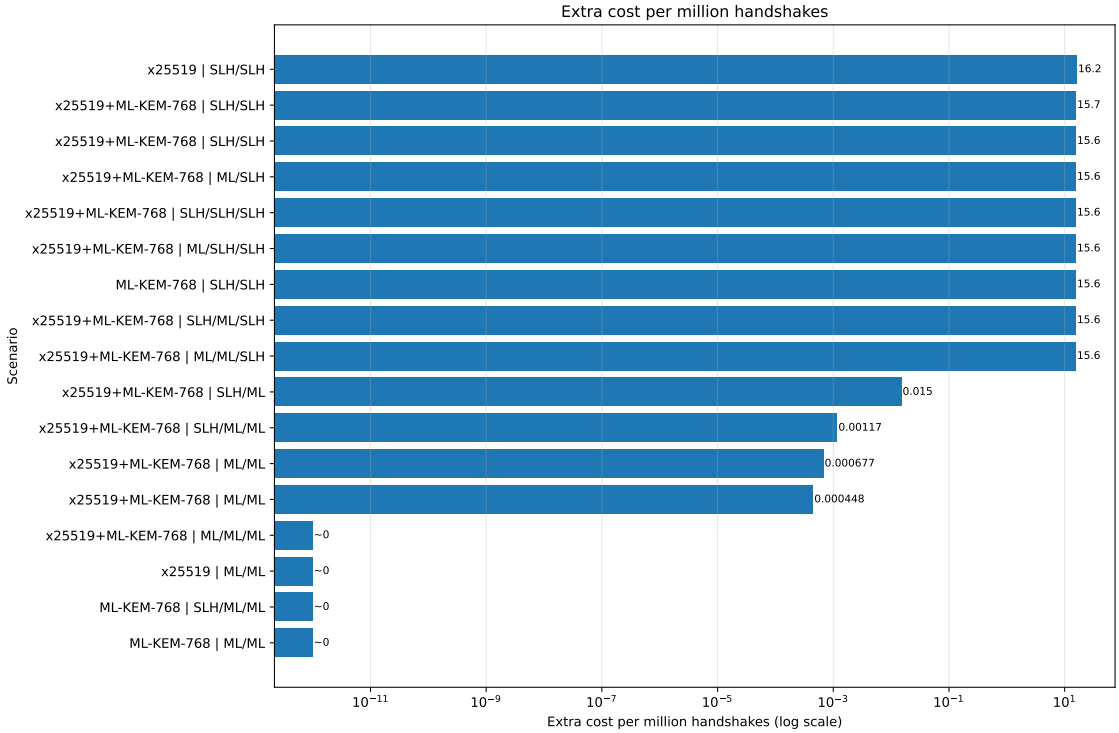


Figure 19: Extra cost per million handshakes.

front-end. This is consistent with the bounded-penalty interpretation developed in the earlier sections.

The leaf-SLH class behaves very differently. Its median extra monthly cost rises from approximately 46.8114 in the small internal service class to about 4681.1385 in the medium API class and about 46811.3845 in the high-volume front-end class. The scaling pattern itself is unsurprising, since the model is linear in traffic. In practical terms, a decision that may look merely expensive in a low-volume setting becomes economically severe in an edge-facing or high-throughput interactive deployment.

9.5 Operational plausibility of hierarchy strategies

The final step is to restate the results in explicitly operational terms. Earlier in the paper, the strategy space was described through four classes: *Reasonable*, *Penalized but plausible*, *Operationally problematic*, and *Unsuitable for interactive TLS front-end*.

The all-ML strategies remain firmly in the *Reasonable* class. They are low-latency, structurally balanced, capacity-stable, and economically aligned with the baseline. The `root-SLH / intermediate-ML / leaf-ML` strategy belongs in *Penalized but plausible*. It introduces a real cost in latency, capacity, and deployment overhead, but remains within a bounded interactive regime.

The leaf-SLH strategies do not belong to the same continuum. Their latency plateau, server-bound compute profile, capacity collapse, infrastructure multipliers, and cost expansion all point in the same direction. In low-volume settings they may still be tolerated for reasons outside

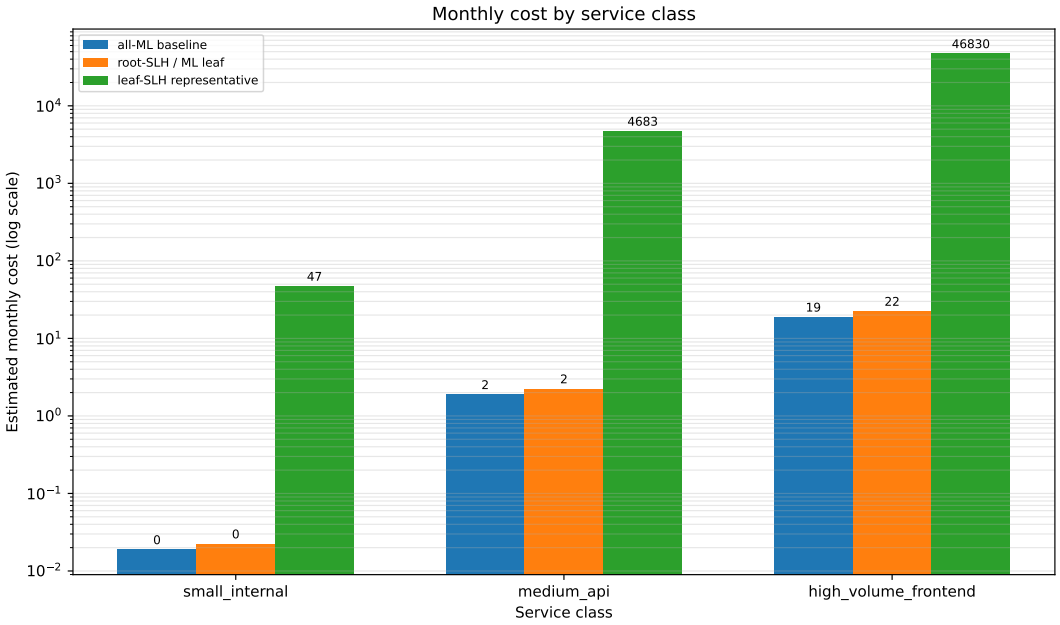


Figure 20: Monthly cost by service class.

performance, but from the perspective of interactive service engineering they belong in the *Unsuitable* class. The heavy regime is operationally disqualifying for front-end TLS use.

10 Threats to Validity and Limitations

The conclusions of this paper are bounded by the experimental setting studied here. Their scope is a controlled local TLS laboratory, a specific implementation stack, and a defined set of post-quantum certificate-hierarchy scenarios. Several limitations therefore constrain the interpretation of the results.

10.1 Local execution environment

All measurements were collected in a controlled local environment, with the final dataset generated on bare metal rather than over a wide-area network. This design reduces network noise and allows certificate-path effects, chain exposure, and client/server cryptographic burden to be isolated more cleanly. It also leaves WAN latency, Internet path variability, queueing under production concurrency, and deployment-specific interactions with middleboxes and edge infrastructure outside the measurement setting.

Accordingly, the latency values reported here should be interpreted as comparative measurements within the tested stack, rather than as direct predictions of end-user Internet latency under arbitrary operating conditions.

10.2 Single implementation stack

The study is conducted on a specific implementation stack, namely OpenSSL 3 together with oqsprovider and liboqs. The results are implementation-dependent: different TLS libraries, certificate-handling paths, provider integrations, or post-quantum implementations could shift absolute values and potentially alter some lower-level trade-offs.

Table 11: Infrastructure capacity summary.

scenario_id	kex_mode	depth	handshakes_per_core_second	handshakes_per_vcpu_hour	capacity_retained_vs_baseline
mlkem768__ml_root__ml_leaf	pure_pqc	2	1,940.97	6,987,482.70	1.0906
mlkem768__slh_root__ml_int__ml_leaf	pure_pqc	3	1,896.59	6,827,725.12	1.0657
x25519__leaf_mldsa65	classical	2	1,891.31	6,808,716.67	1.0627
x25519mlkem768__ml_root__ml_int__ml_leaf	hybrid	3	1,779.68	6,406,856.76	1.0000
x25519mlkem768__ml_root__ml_leaf	hybrid	2	1,660.56	5,978,030.74	0.9331
x25519mlkem768__leaf_mldsa65	hybrid	2	1,605.53	5,779,919.92	0.9021
x25519mlkem768__slh_root__ml_int__ml_leaf	hybrid	3	1,498.86	5,395,885.64	0.8422
x25519mlkem768__slh_root__ml_leaf	hybrid	2	522.29	1,880,240.19	0.2935
x25519mlkem768__ml_root__ml_int__slh_leaf	hybrid	3	0.71	2,569.28	0.0004
x25519mlkem768__slh_root__ml_int__slh_leaf	hybrid	3	0.71	2,568.04	0.0004
mlkem768__slh_root__slh_leaf	pure_pqc	2	0.71	2,566.12	0.0004
x25519mlkem768__ml_root__slh_int__slh_leaf	hybrid	3	0.71	2,565.73	0.0004
x25519mlkem768__slh_root__slh_int__slh_leaf	hybrid	3	0.71	2,562.45	0.0004
x25519mlkem768__ml_root__slh_leaf	hybrid	2	0.71	2,562.41	0.0004
x25519mlkem768__slh_root__slh_leaf	hybrid	2	0.71	2,562.12	0.0004
x25519mlkem768__leaf_slhdsashake192s	hybrid	2	0.71	2,550.76	0.0004
x25519__leaf_slhdsashake192s	classical	2	0.68	2,461.91	0.0004

scenario_id	kex_mode	infrastructure_multiplier_needed	conceptual_perf_group
mlkem768__ml_root__ml_leaf	pure_pqc	0.9169	all_ml
mlkem768__slh_root__ml_int__ml_leaf	pure_pqc	0.9384	root_slh_leaf_ml
x25519__leaf_mldsa65	classical	0.9410	all_ml
x25519mlkem768__ml_root__ml_int__ml_leaf	hybrid	1.0000	all_ml
x25519mlkem768__ml_root__ml_leaf	hybrid	1.0717	all_ml
x25519mlkem768__leaf_mldsa65	hybrid	1.1085	all_ml
x25519mlkem768__slh_root__ml_int__ml_leaf	hybrid	1.1874	root_slh_leaf_ml
x25519mlkem768__slh_root__ml_leaf	hybrid	3.4075	root_slh_leaf_ml
x25519mlkem768__ml_root__ml_int__slh_leaf	hybrid	2493.6366	leaf_slh
x25519mlkem768__slh_root__ml_int__slh_leaf	hybrid	2494.8466	leaf_slh
mlkem768__slh_root__slh_leaf	pure_pqc	2496.7094	leaf_slh
x25519mlkem768__ml_root__slh_int__slh_leaf	hybrid	2497.0893	leaf_slh
x25519mlkem768__slh_root__slh_int__slh_leaf	hybrid	2500.2820	leaf_slh
x25519mlkem768__ml_root__slh_leaf	hybrid	2500.3237	leaf_slh
x25519mlkem768__slh_root__slh_leaf	hybrid	2500.6093	leaf_slh
x25519mlkem768__leaf_slhdsashake192s	hybrid	2511.7455	leaf_slh
x25519__leaf_slhdsashake192s	classical	2602.3964	leaf_slh

For that reason, the strongest claims of the paper are structural. Within a real-stack setting, hierarchy-sensitive signature placement can dominate the operational cost of post-quantum TLS authentication to an extent that flat algorithm comparisons fail to capture. The paper does not assert that every TLS implementation would produce exactly the same ratios or absolute timings.

10.3 Unequal run counts and scope of inference

Run counts were not uniform across scenarios. Fast scenarios were executed more extensively, whereas extremely heavy scenarios, especially those involving SLH-DSA in the leaf, were run fewer times because each handshake was much more expensive to execute.

A larger sample size improves the stability of descriptive estimates for a given scenario in the tested environment. It does not broaden the scope of the claims beyond that environment, nor does it remove implementation-specific behavior, measurement dependence across repeated runs, or other stack-level sources of bias.

The central findings of the paper do not depend on resolving small deltas among near-equivalent configurations. They rest on sharply separated operational regimes: sub-millisecond and low-millisecond all-ML or upper-layer-SLH cases on the one hand, and second-scale leaf-SLH cases on the other. Those separations remain visible in mean latency, p95 latency, transport volume, and server-side task-clock.

Table 12: Economic summary by scenario/class.

scenario_id	conceptual economic class	server CPU seconds per handshake	handshakes per server CPU second	handshakes per server CPU hour	capacity retained vs baseline	capacity loss vs baseline (%)	infrastructure multiplier needed
x25519mlkem768__ml_root__ml_int__ml_leaf	all_ml	0.0006	1779.6824	6406856.7605	1.0000	0.0000	1.0000
x25519mlkem768__slh_root__ml_int__ml_leaf	root_slh_leaf_ml	0.0007	1498.8571	5395885.6372	0.8422	15.7795	1.1874
x25519mlkem768__slh_root__slh_int__slh_leaf	leaf_slh	1.4049	0.7118	2562.4537	0.0004	99.9600	2500.2820
x25519mlkem768__ml_root__slh_leaf	leaf_slh	1.4049	0.7118	2562.4109	0.0004	99.9600	2500.3237
x25519__leaf_slhdsashake192s	leaf_slh	1.4623	0.6839	2461.9066	0.0004	99.9616	2602.3964

scenario_id	conceptual economic class	CPU hours per million handshakes	cost per million default	extra cost per million default	cost multiplier vs baseline
x25519mlkem768__ml_root__ml_int__ml_leaf	all_ml	0.1561	0.0062	0.0000	1.0000
x25519mlkem768__slh_root__ml_int__ml_leaf	root_slh_leaf_ml	0.1853	0.0074	0.0012	1.1874
x25519mlkem768__slh_root__slh_int__slh_leaf	leaf_slh	390.2510	15.6100	15.6038	2500.2820
x25519mlkem768__ml_root__slh_leaf	leaf_slh	390.2575	15.6103	15.6041	2500.3237
x25519__leaf_slhdsashake192s	leaf_slh	406.1893	16.2476	16.2413	2602.3964

10.4 Semantics of observed chain exposure

A recurring methodological subtlety throughout the paper is that the effective chain observed by the client is not always equivalent to the naive logical PKI topology. In the dataset analyzed here, `chain_len_unique = 2` does not imply that the hierarchy is logically depth 2, and `served_chain_der_bytes` does not carry fully homogeneous semantics across all topology classes.

This imposes interpretive discipline on chain-level analysis. For that reason, the paper relies primarily on `bytes_read_mean` and `chain_bytes_unique` for strict cross-depth transport reasoning, and treats `served_chain_der_bytes` with explicit caution. As a result, some chain-level conclusions are stated in empirical terms tied to observed exposure, rather than in idealized PKI-topology terms.

10.5 Perf as proxy rather than full function tracing

The client/server performance-counter analysis is one of the strongest components of the paper, but it remains aggregate evidence rather than full internal function tracing. The results show clearly that leaf-SLH scenarios are overwhelmingly server-side and nearly compute-bound at the server, but they do not by themselves identify a unique internal function, call path, or provider-level routine as the exclusive source of cost. The performance-counter evidence should be interpreted as regime-level causal support rather than as proof of a uniquely identified internal mechanism.

10.6 Economic model as derived interpretation

The economic section of the paper is intentionally framed as an operational interpretation derived from measured server-side compute cost. The service classes, pricing assumptions, and cost-per-million-handshakes figures are illustrative and parameterized rather than tied to a specific commercial contract, cloud provider invoice, or production deployment profile. The resulting

Table 13: Service-class economic translation.

service class	conceptual economic class	scenarios	mean daily cost	median daily cost	mean extra daily cost vs baseline	median extra daily cost vs baseline	mean monthly cost (30d)	median monthly cost (30d)
high_volume_frontend	all_ml	5	0.6291	0.6243	0.0048	0.0000	18.8726	18.7299
high_volume_frontend	leaf_slh	9	1567.6010	1561.0038	1566.9767	1560.3795	47028.0298	46830.1144
high_volume_frontend	root_slh_leaf_ml	3	1.1515	0.7413	0.5272	0.1170	34.5454	22.2392
medium_api	all_ml	5	0.0629	0.0624	0.0005	0.0000	1.8873	1.8730
medium_api	leaf_slh	9	156.7601	156.1004	156.6977	156.0379	4702.8030	4683.0114
medium_api	root_slh_leaf_ml	3	0.1152	0.0741	0.0527	0.0117	3.4545	2.2239
small_internal	all_ml	5	0.0006	0.0006	0.0000	0.0000	0.0189	0.0187
small_internal	leaf_slh	9	1.5676	1.5610	1.5670	1.5604	47.0280	46.8301
small_internal	root_slh_leaf_ml	3	0.0012	0.0007	0.0005	0.0001	0.0345	0.0222

service class	conceptual economic class	scenarios	mean extra monthly cost (30d) vs baseline	median extra monthly cost (30d) vs baseline	mean annual cost (365d)	mean extra annual cost (365d) vs baseline
high_volume_frontend	all_ml	5	0.1427	0.0000	229.6166	1.7358
high_volume_frontend	leaf_slh	9	47009.2998	46811.3845	572174.3620	571946.4811
high_volume_frontend	root_slh_leaf_ml	3	15.8155	3.5092	420.3024	192.4215
medium_api	all_ml	5	0.0143	0.0000	22.9617	0.1736
medium_api	leaf_slh	9	4700.9300	4681.1385	57217.4362	57194.6481
medium_api	root_slh_leaf_ml	3	1.5815	0.3509	42.0302	19.2422
small_internal	all_ml	5	0.0001	0.0000	0.2296	0.0017
small_internal	leaf_slh	9	47.0093	46.8114	572.1744	571.9465
small_internal	root_slh_leaf_ml	3	0.0158	0.0035	0.4203	0.1924

quantities therefore express the scale of the measured compute burden under the stated model, not the exact future bill of any particular organization.

11 Conclusion

11.1 Main finding

The main finding of this paper is that post-quantum migration in TLS 1.3 is best understood as a certificate-hierarchy design problem rather than as a flat comparison between signature algorithms. Across all four campaigns, the most important practical distinction is whether SLH-DSA reaches the handshake-exposed server leaf.

Once SLH-DSA is placed in that position, the authentication path enters a qualitatively different regime. Mean latency moves from the sub-millisecond or low-millisecond region into the ~1.4-second range, server-side active compute becomes almost identical to end-to-end elapsed time, retained capacity collapses, and infrastructure multipliers rise by orders of magnitude. By contrast, scenarios that confine SLH-DSA to upper trust layers while preserving ML-DSA in the interactive leaf stay within a bounded and materially more plausible regime.

The paper therefore supports a precise conclusion. In TLS, the operational meaning of a post-quantum signature family is determined by its presence in the hierarchy and by its placement within the live authentication path.

11.2 Implications for certificate hierarchy design

This conclusion has direct implications for the design of post-quantum certificate hierarchies. A migration strategy should be evaluated in terms of standardization status, primitive-level conservatism, conceptual uniformity across the PKI, handshake exposure, and the location where cryptographic burden is paid.

The results suggest that uniform post-quantum placement across the entire hierarchy is not necessarily the most rational deployment strategy. The data support a more selective approach in which heavier signature families may be tolerated in upper trust layers while the interactive server leaf is protected from the regime-defining cost of SLH-DSA. Within the design space

studied here, `root-SLH / intermediate-ML / leaf-ML` emerges as a materially penalized but still plausible strategy, whereas leaf-SLH configurations occupy a different operational class.

11.3 What remains open

Several lines of work remain open. First, the present study uses a single implementation stack. Repeating the same hierarchy-sensitive analysis across other TLS libraries and post-quantum integrations would strengthen confidence in the generality of the structural findings. Second, the present work focuses on full handshakes rather than resumed sessions. Session resumption may alter the practical weight of certificate-path cost in some deployment settings.

Finally, the broader post-quantum transition raises a more general design question that goes beyond the specific scenarios measured here: how should protocol designers and operators distribute cryptographic conservatism across the certification hierarchy when different positions in that hierarchy carry radically different operational meaning? The present paper does not settle that question in general, but it shows that answering it requires more than flat primitive comparison.

References

[1] Sharon Boeyen et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. May 2008. DOI: 10.17487/RFC5280. URL: https://www.rfc-editor.org/info/rfc5280.

[2] Deirdre Connolly. *ML-KEM Post-Quantum Key Agreement for TLS 1.3*. Internet-Draft draft-ietf-tls-mlkem-07. Work in Progress. Internet Engineering Task Force, Feb. 2026. 11 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-tls-mlkem/07/.

[3] Jose A. Montenegro, Ruben Rios, and Javier Lopez-Cerezo. "A Performance Evaluation Framework for Post-Quantum TLS". In: *Future Generation Computer Systems* 175.108062 (Feb. 2026), pp. 1–14. ISSN: 0167-739X. DOI: 10.1016/j.future.2025.108062. URL: https://doi.org/10.1016/j.future.2025.108062.

[4] National Institute of Standards and Technology. *Module-Lattice-Based Digital Signature Standard*. Federal Information Processing Standards Publication NIST FIPS 204. Gaithersburg, MD: National Institute of Standards and Technology, U.S. Department of Commerce, Aug. 2024. DOI: 10.6028/NIST.FIPS.204. URL: https://doi.org/10.6028/NIST.FIPS.204.

[5] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Federal Information Processing Standards Publication NIST FIPS 203. Gaithersburg, MD: National Institute of Standards and Technology, U.S. Department of Commerce, Aug. 2024. DOI: 10.6028/NIST.FIPS.203. URL: https://doi.org/10.6028/NIST.FIPS.203.

[6] National Institute of Standards and Technology. *Stateless Hash-Based Digital Signature Standard*. Federal Information Processing Standards Publication NIST FIPS 205. Gaithersburg, MD: National Institute of Standards and Technology, U.S. Department of Commerce, Aug. 2024. DOI: 10.6028/NIST.FIPS.205. URL: https://doi.org/10.6028/NIST.FIPS.205.

[7] Open Quantum Safe Project. *liboqs*. https://openquantumsafe.org/liboqs/. Accessed: 2026-05-19.

[8] Open Quantum Safe Project. *oqs-provider: Open Quantum Safe Provider for OpenSSL 3*. https://github.com/open-quantum-safe/oqs-provider. Accessed: 2026-05-19.

[9] Sebastian Paul et al. “Mixed Certificate Chains for the Transition to Post-Quantum Authentication in TLS 1.3”. In: *ASIA CCS '22: Proceedings of the 2022 ACM Asia Conference on Computer and Communications Security*. Association for Computing Machinery, May 2022, pp. 727–740. DOI: 10.1145/3488932.3497755. URL: https://doi.org/10.1145/3488932.3497755.

[10] Tirumaleswar Reddy.K and Hannes Tschofenig. *Post-Quantum Cryptography Recommendations for TLS-based Applications*. Internet-Draft draft-ietf-uta-pqc-app-01. Work in Progress. Internet Engineering Task Force, Feb. 2026. 23 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-uta-pqc-app/01/.

[11] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Aug. 2018. DOI: 10.17487/RFC8446. URL: https://www.rfc-editor.org/info/rfc8446.

[12] Dimitrios Sikeridis, Panos Kampanakis, and Michael Devetsikiotis. “Post-Quantum Authentication in TLS 1.3: A Performance Study”. In: *Proceedings of the Network and Distributed System Security Symposium (NDSS 2020)*. Feb. 2020. DOI: 10.14722/ndss.2020.24203. URL: https://www.ndss-symposium.org/wp-content/uploads/2020/02/24203.pdf.

[13] Douglas Stebila, Scott Fluhrer, and Shay Gueron. *Hybrid Key Exchange in TLS 1.3*. Internet-Draft draft-ietf-tls-hybrid-design-16. Work in Progress. Internet Engineering Task Force, Sept. 2025. 23 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/16/.

A Scenario Inventory

This appendix provides the full experimental inventory used in the study. The main text discusses only those scenario distinctions needed for causal and operational interpretation, whereas this appendix preserves the broader inventory for traceability and reproducibility.

Scenario identifiers are constructed compositionally. They encode the key-establishment mode together with the signature-family placement across the logical certificate hierarchy. In the naming convention used throughout the paper, `root`, `int`, and `leaf` denote certificate positions in the hierarchy, while labels such as `ml` and `slh` denote the corresponding signature-family assignment. Depth-2 scenarios omit the intermediate position by construction.

Table 14: Experimental inventory and scenario matrix.

Campaign	Scenario	TLS group	Root	Intermediate	Leaf	Depth	Runs
A	x25519__leaf_mldsa65	x25519	ML-DSA	—	ML-DSA	2	10000
A	x25519__leaf_slhdsashake192s	x25519	SLH-DSA	—	SLH-DSA	2	300
A	x25519mlkem768__leaf_mldsa65	x25519mlkem768	ML-DSA	—	ML-DSA	2	10000
A	x25519mlkem768__leaf_slhdsashake192s	x25519mlkem768	SLH-DSA	—	SLH-DSA	2	300
B	x25519mlkem768__ml_root__ml_int__ml_leaf	x25519mlkem768	ML-DSA	ML-DSA	ML-DSA	3	10000
B	x25519mlkem768__ml_root__ml_int__slh_leaf	x25519mlkem768	ML-DSA	ML-DSA	SLH-DSA	3	300
B	x25519mlkem768__ml_root__slh_int__slh_leaf	x25519mlkem768	ML-DSA	SLH-DSA	SLH-DSA	3	300
B	x25519mlkem768__slh_root__ml_int__ml_leaf	x25519mlkem768	SLH-DSA	ML-DSA	ML-DSA	3	10000
B	x25519mlkem768__slh_root__ml_int__slh_leaf	x25519mlkem768	SLH-DSA	ML-DSA	SLH-DSA	3	300
B	x25519mlkem768__slh_root__slh_int__slh_leaf	x25519mlkem768	SLH-DSA	SLH-DSA	SLH-DSA	3	300
C	x25519mlkem768__ml_root__ml_leaf	x25519mlkem768	ML-DSA	—	ML-DSA	2	10000
C	x25519mlkem768__ml_root__slh_leaf	x25519mlkem768	ML-DSA	—	SLH-DSA	2	300
C	x25519mlkem768__slh_root__ml_leaf	x25519mlkem768	SLH-DSA	—	ML-DSA	2	10000
C	x25519mlkem768__slh_root__slh_leaf	x25519mlkem768	SLH-DSA	—	SLH-DSA	2	300
D	mlkem768__ml_root__ml_leaf	mlkem768	ML-DSA	—	ML-DSA	2	10000
D	mlkem768__slh_root__ml_int__ml_leaf	mlkem768	SLH-DSA	ML-DSA	ML-DSA	3	10000
D	mlkem768__slh_root__slh_leaf	mlkem768	SLH-DSA	—	SLH-DSA	2	300