Baltic J. Modern Computing, Vol. 13 (2025), No. 4, pp. 862–884
https://doi.org/10.22364/bjmc.2025.13.4.06

Practical Assessment of the SSH Services' Transition to Post-Quantum Cryptography

Simona ZAVACKĖ, Linas BUKAUSKAS

Institute of Computer Science, Vilnius University, Vilnius, Lithuania

simona.zavacke@mif.stud.vu.lt, linas.bukauskas@mif.vu.lt

ORCID 0009-0007-5910-5658, ORCID 0000-0002-9781-9690

Abstract. Quantum computing poses a critical threat to classical public-key cryptography, driving the adoption of post-quantum cryptography (PQC). While PQC performance has been extensively benchmarked in TLS, empirical studies on the Secure Shell (SSH) protocol remain limited. This paper evaluates the integration of PQC and hybrid (classical+PQC) mechanisms into SSH using an OQS-OpenSSH 8.9 prototype with liboqs support. We measured handshake latency, session size, and Secure Copy Protocol (SCP) transfer performance across standardized algorithms (CRYSTALS-Kyber, CRYSTALS Dilithium, Sphincs+), Falcon, and hybrid configurations. Experiments on heterogeneous legacy hardware under realistic LAN conditions were validated through Wireshark captures and protocol checklists. The results show that Falcon yields the smallest handshake sizes, Dilithium offers balanced and stable performance, Kyber scales predictably with NIST security levels, and Sphincs+ incurs significant overhead. Hybrid modes add limited cost while retaining classical trust anchors. These findings extend SSH PQC research beyond handshake analysis, offering deployment-oriented guidance for quantum-safe migration.

Keywords: post-quantum cryptography, hybrid PQC, transition to PQC, SSH, ML-KEM, Kyber, ML-DSA, SLH-DSA, Falcon, NIST security levels

1 Introduction

The rapid development of quantum computing represents a credible threat to existing asymmetric cryptographic methods. Schemes such as RSA, DSA, and Elliptic Curve Cryptography (ECC) rely on number-theoretic problems whose security assumptions collapse in the presence of quantum adversaries. In response to this, the National Institute of Standards and Technology (NIST) finalized the first Post-Quantum Cryptography (PQC) standards on August 13, 2024. These standards, published as FIPS 203, FIPS 204, and FIPS 205, are designed to protect against attacks from quantum computers (Jacob W. S. Schneider, 2024; National Institute of Standards and Technology, 2024e,a):

- ML-KEM (FIPS 203), a module lattice-based key encapsulation mechanism derived from CRYSTALS-*Kyber* algorithm is for key exchange (National Institute of Standards and Technology, 2024b);
- ML-DSA (FIPS 204), a module lattice-based digital signature algorithm based on CRYSTALS *Dilithium* algorithm (National Institute of Standards and Technology, 2024c);
- SLH-DSA (FIPS 205), a stateless hash-based digital signature scheme derived from *Sphincs+* algorithm, as a backup in case ML-DSA is compromised (National Institute of Standards and Technology, 2024d).

Additional candidates, such as Falcon, remain under evaluation for inclusion in future standards (e.g., as FIPS 206, FN-DSA). Falcon is a lattice-based digital signature algorithm that leverages Fast Fourier Transforms (FFT) to optimize the efficiency and speed of key generation, signing, and verification (Prest et al., 2020). While its compact signatures and high verification performance make it an attractive option, Falcon's reliance on discrete Gaussian sampling introduces implementation complexity and potential side-channel risks. These factors, along with the need for further cryptanalysis and validation of secure, constant-time implementations, are among the reasons why Falcon has not yet completed the full standardization process (National Institute of Standards and Technology, 2022b; Pierre-Alain et al., 2024).

To ensure continuity of trust during the transition to PQC, some organizations and researchers have explored hybrid cryptographic constructions, which combine classical and PQC components to provide security guarantees (Bindel et al., 2019; Crockett et al., 2019; Rossi, 2021). For example, such approaches are reflected in X.509 hybrid certificate prototypes (Nina et al., 2019), hybrid TLS 1.3 key exchanges (Stebila et al., 2021), and IETF drafts on hybrid signature mechanisms (Ounsworth and Pala, 2024).

Despite significant progress in developing PQC algorithms and advancing hybrid approaches, the most significant challenge is ensuring that new cryptographic algorithms can be integrated seamlessly into existing systems, applications, and protocols. The key difficulty lies in validating the practical feasibility of PQC schemes: behavior under real operational conditions, compatibility with existing protocols, the magnitude of their computational burden, and their sensitivity to subtle integration flaws. Real-life experimentation is essential to assess not only the computational cost of PQC primitives but also their resilience to implementation-specific pitfalls such as timing leaks, resource exhaustion, or protocol-level incompatibilities. Without such empirical grounding, recommendations for PQC migration remain speculative and potentially misleading (Ott et al., 2019; Alnahawi et al., 2021).

Running a pilot project to evaluate the performance and security of new algorithms in real-world scenarios is a critical step in understanding how these PQC cryptographic systems behave under operational conditions. Real-world performance evaluation enables the identification of the most suitable algorithm variant and the fine-tuning of performance parameters (Mosca, 2018; Kreutzer et al., 2018; Barker et al., 2021; Joseph et al., 2024). This process helps validate the practical feasibility of PQC schemes and supports a smoother migration to a quantum-safe state.

The objective of this research is to evaluate the integration of PQC and its hybrid implementation into the Secure Shell (SSH) protocol. By developing a functioning

prototype based on OpenSSH and the liboqs framework, this study investigates the performance, compatibility, and practical deployment considerations of PQC algorithms as ML-KEM, ML-DSA, SLH-DSA, Falcon, and selected hybrid configurations.

The contributions of this work are threefold:

1. Integrating and evaluating PQC and its hybrid implementation in OpenSSH using liboqs.
2. Measuring handshake size and latency for various algorithm combinations in a controlled but realistic environment;
3. Analyzing the trade-offs between security level, session size, and connection time.
4. Recommending optimal algorithm configurations for practical PQC adoption in SSH.

This paper is structured as follows. Section 2 reviews the relevant literature and previous SSH-related PQC evaluations. Section 3 details the prototype environment, algorithms, and measurement methodology. Section 4 presents and interprets the performance data, followed by deeper analysis in Section 5. Section 6 summarizes conclusions and outlines directions for future work.

2 Related Work

PQC research has focused mainly on design and formal security analysis, while research on their practical implementation in protocols remains limited. Prior work on PQC integration in protocols is richest in TLS, where macro-benchmarks and even protocol variants (e.g., KEMTLS) quantify handshake time trade-offs and message size growth (Paquin et al., 2019; Schwabe et al., 2020; Stebila et al., 2021). TLS benchmarks reveal non-trivial PQC impact on handshake latency:

– Paquin et al. (2019) presents one of the first comprehensive integrations of PQC into mainstream TLS libraries. They integrated Kyber and Dilithium into OpenSSL and BoringSSL (TLS 1.3), experimented on virtual machines the client and server processes. Results demonstrated that while post-quantum public-key operations dominate median TLS handshake latency on fast, reliable links, on lossy networks ($\geq$3–5% packet loss), larger PQC message sizes become the primary bottleneck—and as application payloads increase, the relative cost of the PQC handshake rapidly diminishes.
– Schwabe et al. (2020) propose KEMTLS, removing signature rounds in TLS, and show optimized KEM-only handshakes cut latency by up to 30 %.

Although our focus is on SSH, we review PQC benchmarking in TLS because the two protocols share cryptographic primitives and migration concerns. TLS studies provide valuable baselines and methodological lessons, while also highlighting the distinctive aspects of SSH. In particular, TLS results emphasize handshake latency in short-lived connections, whereas SSH's session-oriented design raises different challenges such as sustained throughput and multi-client concurrency. Reviewing TLS benchmarks, therefore, helps situate our contribution within the broader landscape of PQC deployment studies.

SSH has credible but comparatively fewer empirical studies, mostly handshake-oriented. In particular, the handshake latency, session size, and throughput impacts of PQC remain underexplored outside of controlled laboratory settings:

- Crockett, Paquin, and Stebila (Crockett et al., 2019) demonstrated how PQC and hybrid key exchange could be integrated into SSH (and TLS) as proof-of-concept, using liboqs primitives and adapting OpenSSH code paths. Their emphasis was on design feasibility, not full-scale performance benchmarking.
- Sikeridis et al. (2020b) implemented hybrid ECDHE+PQC key exchanges (Kyber, Dilithium, SPHINCS+) in both TLS 1.3 and OpenSSH, demonstrating that PQC integration incurs handshake latency overheads ranging from just 0.5 % (Kyber) up to 50 % (SPHINCS+) and shows that simple TCP initial-window tuning can halve the perceived slowdown. These findings suggest that both cryptographic and network-level considerations are critical in PQC deployment. This research provided valuable early measurements of PQC integration into SSH, but their focus remained primarily on handshake costs, not sustained session or file transfer performance.

Formal analyses of hybrid SSH protocol correctness and security under hybrid models have been treated by:

- Crockett et al. (2019), who implemented and discussed protocol-level changes for hybrid PQC key exchange in OpenSSH 7.9.
- Tran et al. (2024), using symbolic methods (CafeOBJ) to verify that classical and PQC components interoperate securely.
- Kampanakis et al. (2020) represents one of the first standardization-oriented efforts to define how SSH can negotiate and fall back safely between classical and PQC key exchange algorithms. Their design patterns inform the hybrid architecture evaluated in this paper.

Our work complements these efforts by extending the analysis to full SSH sessions on real hardware, including file transfer workloads and hybrid configurations. Whereas most prior studies combine performance metrics with CPU profiling or formal analysis often limited to single-client PQC implementations in virtualized settings, this work focuses exclusively on session-level performance (handshake latency, session size, and throughput). We evaluate PQC in a uniform OpenSSH environment under realistic client–server loads, deployment–focused recommendations. In doing so, this study contributes empirical evidence that helps close the "SSH in the wild" gap by:

- Incorporating Falcon into SSH performance benchmarking.
- Testing on heterogeneous, legacy hardware representative of operational infrastructures.
- Quantifying handshake size and latency across PQC and hybrid configurations.
- Evaluating large file transfers to capture end-to-end SSH workload performance.
- Providing a balanced assessment of algorithm choices, weighing operational efficiency against required security assurance levels.

Table 1 summarizes how our work extends prior research.

Table 1: Comparison of selected SSH-related PQC studies

Study	Algorithms	Highlights
Crockett et al. (2019)	Kyber, Dilithium, SPHINCS+	First hybrid SSH in OpenSSH 7.9. No Falcon. Limited performance analysis.
Sikeridis et al. (2020a)	NIST Round 2/3	Latency testing under PQC; observed 0.5–50% delay; TCP tuning mitigates impact.
Stebila et al. (2021)	Kyber + ECC	Designed hybrid KEM groups for TLS 1.3; proposed hybridization strategies.
This work	Kyber, Dilithium, Falcon, SPHINCS+	Full SSH implementation on real hardware. Includes Falcon. Benchmarked handshakes and file transfers (SCP). Performance is evaluated in terms of security level.

3 Methods

We designed a replicable prototype method for the realistic operational conditions of SSH services and developed a prototype to empirically evaluate and identify the most suitable PQC algorithms and hybrid implementations for SSH-based communication. SSH works as an application and establishes automated connections.

3.1 Sandbox network architecture

To simulate real-world operating conditions, an evaluation test environment was created using legacy equipment representing a typical small business environment. Figure 1 illustrates the sandbox topology, where the client and server are interconnected through a Local Area Network. Experimental setup:

- Client: Ubuntu 20.04.5 LTS, Intel i5-4570S CPU (4 cores, 2.9 GHz), 16 GB RAM.
- Server: Ubuntu 22.04.1 LTS, AMD A8-5500 APU with Radeon HD Graphics (4 cores, 2.14 GHz), 16 GB RAM.
- Network infrastructure: TP-Link TL-SF1005D desktop switch (100 Mbps Ethernet), realistically limiting bandwidth. Network speed validated at 94 Mbps via `iperf`.

The prototype leveraged OpenSSH 8.9 (Git tag V_8_9_P1), integrating PQC through OQS-OpenSSH, a fork designed explicitly for quantum-resistant cryptography using liboqs version 0.7.2. libraries[1]. Initially developed for isolated environments (where client-server interactions were confined to a single virtual machine), the prototype required modifications to the SSH configuration files to enable automated connections between client and server instances running on separate physical devices. SSH was configured explicitly over IPv4, and TCP window scaling extensions were enabled to reflect common operational practices for optimized network data transmission.

[1] The prototype was made in 2022 November, OQS-OpenSSH snapshot 2022-08 was used`https://github.com/open-quantum-safe/openssh`. The old names of algorithms are used, as they were tested at that time, and there are some minor differences with standardised versions.

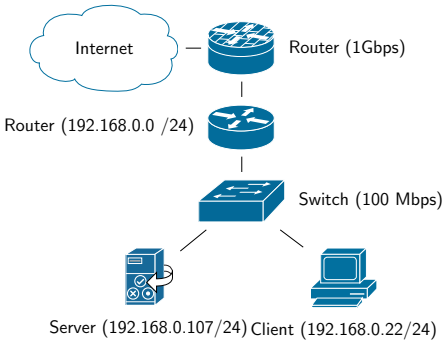

Fig. 1: Prototyping SSH: network test-bed

3.2 Algorithms under test

For comparative benchmarking, the prototype first establishes a *classical* cryptographic baseline. This baseline uses Ed25519 digital signatures[2] in combination with Elliptic Curve Diffie–Hellman (ECDH) key exchange across three standardized NIST recommended curves: `ecdh-nistp256`, `ecdh-nistp384`, and `ecdh-nistp521`.[3]

Table 2: SSH session performance with classical algorithms: Ed25519 signatures and ECDH key exchange

Key Exchange with Ed25519	**Time (ms)**	**Size (bytes)**	**NIST Security**
ECDH-nistp256	459	34,540	Level 1
ECDH-nistp384	475	34,700	Level 3
ECDH-nistp521	465	34,942	Level 5
Average	**466**	**34,727**	–

As shown in Table 2, this configuration achieves an average SSH handshake size of **34,727 bytes** and an average handshake latency of **466 ms**. These values serve as the primary reference point for evaluating the performance and practicality of PQC and hybrid implementations.

[2] Ed25519 is a high-performance Edwards-curve signature scheme designed for fast verification, short keys, and strong security guarantees (Bernstein et al., 2012). For more: `https://ed25519.cr.yp.to`, accessed 2025-08-05.

[3] ECDH is supported in SSH using the NIST recommended prime field curves `nistp256`, `nistp384`, and `nistp521`, implemented in OpenSSH as `ecdh-nistp256`, `ecdh-nistp384`, and `ecdh-nistp521`. RFC 5656 specifies all three for use in SSH key exchange (IETF SSH Working Group, 2009), while NIST SP 800-57 Part 3 designates `nistp256` and `nistp384` as mandatory-to-implement for federal systems, with `nistp521` available as an additional high-security option (Barker et al., 2015).

In PQC-based experiments, the ECDH is replaced by the *Kyber* Key Encapsulation Mechanism (KEM). Table 3 lists the six Kyber parameter sets evaluated. While all generate the same 32-byte shared secret, their key and ciphertext sizes differ as per the NIST security level. `Important note:` Kyber-90s is not standardized by NIST and is included here for comparative performance analysis only. For hybrid implementations, Kyber is combined with the corresponding ECDH parameters[4] to retain classical security alongside PQC resistance.

Table 3: PQC Kyber parameter sets

Parameter set	Public key size (bytes)	Secret key size (bytes)	Ciphertext size (bytes)	NIST Security	Hybrid with
Kyber512	800	1632	768	Level 1	ecdh-nistp256
Kyber512-90s	800	1632	768	Level 1	ecdh-nistp384
Kyber768	1184	2400	1088	Level 3	ecdh-nistp521
Kyber768-90s	1184	2400	1088	Level 3	ecdh-nistp256
Kyber1024	1568	3168	1568	Level 5	ecdh-nistp384
Kyber1024-90s	1568	3168	1568	Level 5	ecdh-nistp521

Authentication in the SSH prototype is performed via PQC digital signatures, as summarized in Table 4, and is tested:

- **Falcon** (lattice-based): offers the smallest signature size among PQC candidates, making it efficient in bandwidth-constrained environments.
- **Dilithium** (lattice-based): provides balanced key sizes and signatures.
- **Sphincs+** (hash-based): delivers strong security, but incurs the largest signature sizes, which may approach SSH packet fragmentation limits.

Table 4: PQC digital signature algorithm parameters

Parameter set	Public key size (bytes)	Secret key size (bytes)	Signature size (bytes)	NIST Security	Hybrid with
Falcon-512	897	1281	690	Level 1	ecdsa-nistp256
Falcon-1024	1793	2305	1330	Level 5	ecdsa-nistp521
Dilithium2-AES	1312	2528	2420	Level 1	ecdsa-nistp256
Dilithium3	1952	4000	3293	Level 3	ecdsa-nistp384
Dilithium5-AES	2592	4864	4595	Level 5	ecdsa-nistp521
Sphincs+ -128f	32	64	17088	Level 1	ecdsa-nistp256
Sphincs+ -192f	48	96	35664	Level 3	ecdsa-nistp384

[4] NIST elliptic curves follow the FIPS 186-5 specification National Institute of Standards and Technology (2023), with curve identifiers such as `nistp256` indicating key length in bits.

From a performance–security trade-off perspective, a higher NIST security level improves long-term resistance but typically increases the size of public keys, ciphertexts, or signatures (National Institute of Standards and Technology, 2016).

These baselines and PQC variants allow us to directly observe how key length, ciphertext size, and signature size translate into measurable SSH handshake performance. By aligning each algorithm with its corresponding NIST security level, these experiments enable the quantification of how the required security margin translates into real-world SSH handshake performance, providing an evidence-based basis for selecting algorithm variants for transition to PQC.

3.3 Evaluation Methodology

We perform repeated Black-Box trials measuring: (i) *Handshake latency* (client connect to finished key exchange and authentication); (ii) *Session-establishment size* (bytes on the wire); and (iii) *SCP transfer* behavior for files of 1 MB, 10 MB, 100 MB, 200 MB and 1024 MB.

Network analysis utilized Wireshark to capture and thoroughly quantify session-related data exchanges during SSH session establishment. For capturing precise timing and event sequencing to ensure comprehensive session validation the detailed SSH protocol steps checklist was created. The primary metrics utilized for evaluating each SSH connection include:

- **Handshake Latency:** The total duration from the initiation of an SSH connection request by the client to the successful completion of the cryptographic handshake, directly influencing user-perceived responsiveness.
- **Session Establishment Data Size:** The volume of network traffic data exchanged during the handshake phase. Lower data volumes reduce network load and enhance scalability, especially in bandwidth-limited environments.
- **Security Level:** Assessed based on NIST's standardized categorization, ranging from Security Level 1 (equivalent to AES-128) to Security Level 5 (equivalent to AES-256) (National Institute of Standards and Technology, 2016). This criterion ensures a balanced evaluation of algorithms, considering both efficiency and relative security assurance.
- **Success criteria.** No downtime; no noticeable performance degradation; and overhead acceptable for typical LAN conditions.

Limitations: Despite efforts toward realism, the sandbox configuration may not fully capture all potential complexities present in large-scale enterprise networks. Results should thus be interpreted with caution concerning generalizability beyond similar organizational contexts.

4 Results

The measurements were collected under the same network, hardware, and OpenSSH and liboqs configurations described in Section 3. Each signature algorithm was tested in both *PQC-only* and *Hybrid* configurations, with handshake session size (in bytes) and handshake time (in milliseconds) recorded for every KEM and signature pairing.

4.1 Performance data: Hybrid implementation

The results obtained from combinations of Hybrid implementations over an SSH session are presented. Table 5 presents data on the size of SSH session establishment in bytes, while Table 6 displays the session time results measured in milliseconds (ms).

Classical *Ed2519* was also combined with all hybrid Kyber to compare against classical ECDH in the SSH protocol. Together, these metrics enable a direct, evidence-based comparison of the bandwidth and latency implications of different Hybrid configurations.

Table 5: Hybrid algorithms combinations: session size (bytes)

Hybrid combinations results (size)	ecdsa-nistp256-Falcon512	ecdsa-nistp521-Falcon1024	ecdsa-nistp384-Dilithium3	ecdsa-nistp256-Dilithium2aes	ecdsa-nistp521-Dilithium5aes	ecdsa-nistp256-Sphincs+-128f	ecdsa-nistp384-Sphincs+-192f	Ed25519
ecdh-nistp256 Kyber512	42384	49958	54942	48730	62048	72130	112076	34892
ecdh-nistp384 Kyber768	42972	50864	55644	49564	62742	72840	112984	35396
ecdh-nistp521 Kyber1024	44172	51860	56786	50714	64016	73784	113532	36546
ecdh-nistp256 Kyber512-90s	**42204**	49826	55146	48738	62122	72014	112290	34636
ecdh-nistp384 Kyber768-90s	42972	50594	56238	50100	63352	73442	113528	35734
ecdh-nistp521 Kyber1024-90s	44502	52256	57256	51184	64552	74386	**114464**	36810

Table 6: Hybrid algorithms combinations: session time (milliseconds)

Hybrid session time (ms)	ecdsa-nistp256-Falcon512	ecdsa-nistp521-Falcon1024	ecdsa-nistp384-Dilithium3	ecdsa-nistp256-Dilithium2aes	ecdsa-nistp521-Dilithium5aes	ecdsa-nistp256-Sphincs+-128f	secdsa-nistp384-Sphincs+-192f	Ed25519
ecdh-nistp256 Kyber512	400	425	482	396	440	401	496	429
ecdh-nistp384 Kyber768	413	416	496	410	474	427	505	406
ecdh-nistp521 Kyber1024	439	404	460	382	425	422	476	415
ecdh-nistp256 Kyber512-90s	**347**	452	464	391	444	423	477	384
ecdh-nistp384 Kyber768-90s	369	453	536	428	503	482	577	421
ecdh-nistp521 Kyber1024-90s	426	477	508	389	483	481	**584**	435

The results of the Hybrid implementation (without *Ed25519*) show the following statistics: the average session size is **64188 bytes**, with an average time of **450 ms**. The medians are 56238 bytes and 440 ms.

4.2 Performance data: PQC implementation

Results of the combination of PQC algorithms over an SSH session are presented.

Table 7: PQC algorithms combinations: session size (bytes)

PQC session size (byte)	Falcon512	Falcon1024	Dilithium3	Dilithium2aes	Dilithium5aes	Sphincs+ -128f	Sphincs+ -192f
Kyber512	**41082**	48350	53614	47468	60358	70926	110748
Kyber768	41786	48930	54450	48172	60930	71498	111452
Kyber1024	43160	49942	55124	49102	62058	72692	111986
Kyber512-90s	41098	48308	53630	47476	60300	70926	110616
Kyber768-90s	41810	49012	54862	48708	61532	72224	111980
Kyber1024-90s	43120	50470	55858	49704	62528	73154	**112844**

Table 8: PQC algorithms combinations: session time (milliseconds)

PQC session time (ms)	Falcon512	Falcon1024	Dilithium3	Dilithium2aes	Dilithium5aes	Sphincsharaka 128fsimple	Sphincsharaka 192frobust
Kyber512	387	390	377	380	385	384	433
Kyber768	404	381	372	383	376	396	423
Kyber1024	390	395	**344**	397	381	363	423
Kyber512-90s	366	376	350	349	379	391	404
Kyber768-90s	388	399	386	410	408	407	**496**
Kyber1024-90s	367	428	390	404	431	456	480

Table 7 presents the size data (in bytes) for SSH session establishment, while Table 8 consists of the session's time results (in milliseconds). The statistical mean indicates that the average session size is **62714** bytes, with an average time of **396** ms. The medians are 54450 bytes and 390 ms.

4.3 Results of file transfer test

This subsection presents the results of SSH file transfers using the Secure Copy Protocol (SCP)[5] in PQC and Hybrid implementations where *Kyber512* operates in conjunction with three signature algorithms: *Sphincs+-128f* (abbreviated as *Sphincs*), *Dilithium2aes* (abbreviated as *Dilithium*) and *Falcon512* (abbreviated as *Falcon*). Files of 1MB, 10MB,

Table 9: File transfer results: Hybrid implementation

Hybrid mode	*File size (MB)*	*Transfer session time (ms)*	*Transfer session size (bytes)*	*Real size (bytes) of file*	*Time without SSH session establishment (ms)*	*Transfer session without SSH establishment and file size (bytes)*
Sphincs	1	435	1170980	1048576	12	50390
Sphincs	10	1187	11059856	10485760	764	502082
Sphincs	100	9354	109957252	104857600	8931	5027638
Sphincs	200	18206	220004306	209715200	17783	10217092
Dilithium	1	397	1148100	1048576	6	50786
Dilithium	10	1209	11043348	10485760	818	508850
Dilithium	100	9254	110009828	104857600	8863	5103490
Dilithium	200	18177	219947906	209715200	17786	10183968
Falcon	1	407	1140840	1048576	60	50060
Falcon	10	1195	11032106	10485760	848	504142
Falcon	100	9269	110003418	104857600	8922	5103614
Falcon	200	18190	219928354	209715200	17843	10170950

Table 10: File transfer results: PQC implementation

Pure PQC mode	*File size (MB)*	*Transfer session time (ms)*	*Transfer session size (bytes)*	*Real size (bytes) of file*	*Time without SSH session establishment (ms)*	*Transfer session without SSH establishment and file size (bytes)*
Sphincs	1	397	1167318	1048576	6	47816
Sphincs	10	1194	11058012	10485760	803	501326
Sphincs	100	9235	110010038	104857600	8844	5081512
Sphincs	200	18205	219933726	209715200	17814	10147600
Dilithium	1	380	1144066	1048576	31	48014
Dilithium	10	1194	11041954	10485760	845	508718
Dilithium	100	9222	110012526	104857600	8873	5107450
Dilithium	200	18185	219929410	209715200	17836	10166734
Falcon	1	384	1136294	1048576	18	46620
Falcon	10	1194	11036235	10485760	828	509377
Falcon	100	9232	109987610	104857600	8866	5088912
Falcon	200	18165	219939392	209715200	17799	10224192

100MB, 200MB, and 1GB were transmitted from the client to the server.

Notably, transferring a 1GB file took the same duration (1 minute and 29 seconds) and the speed of sessions was 11,2 MB/s across all combinations of algorithms in both PQC and Hybrid implementations, making it less relevant for exploring variations. The SSH SCP test results are presented in Table 9 (Hybrid implementation) and Table 10 (PQC implementation), detailing the times and sizes of file transfers.

5 Discussion

5.1 SSH handshake performance: PQC vs Hybrid vs Classical

Figure 2 compares the performance of SSH session establishments in terms of both time and size across three cryptographic configurations: PQC, Hybrid, and classical ECC only. While the classical ECC configuration achieves the smallest session establishment size, its handshake latency is not lower than that of PQC. Both PQC and Hybrid configurations show performance variability depending on the specific key exchange and signature algorithm combinations. Figure 3 presents the same dataset as box plots to highlight

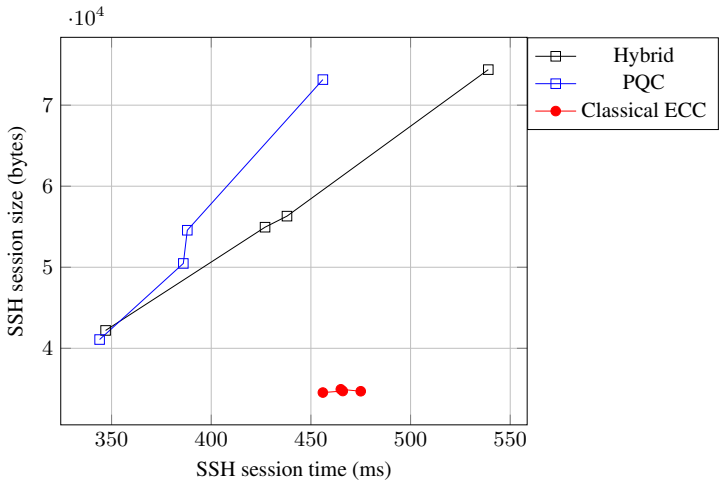

Fig. 2: SSH session establishment time and size for PQC, Hybrid, and Classical ECC configurations.

the statistical distribution of session sizes (left panel) and times (right panel). Each box plot shows the minimum, first quartile, median (represented by a vertical line), third quartile, and maximum values, with outliers indicated as circular markers. From a size perspective, Classical ECC appears as an outlier relative to PQC and Hybrid results due to its significantly smaller handshake sizes. In contrast, handshake time values for Classical ECC fall within the overall spread of Hybrid results. Handshake size results for

[5] SCP is a secure method within SSH for transferring files between a client and a server.

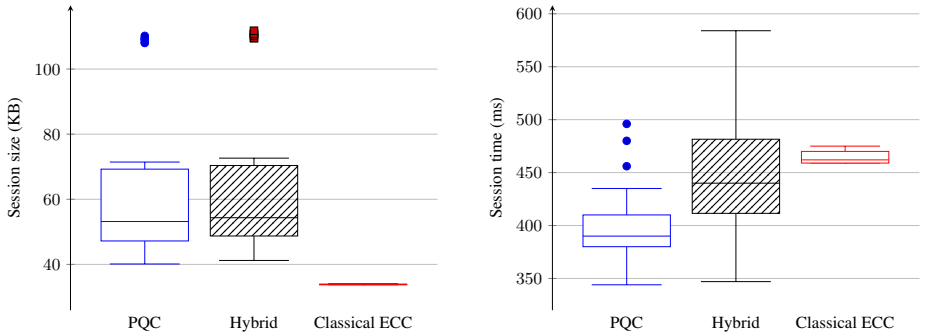

Fig. 3: Box plots of SSH session size and time for PQC, Hybrid, and Classical ECC configurations.

Hybrid and PQC are broadly comparable, showing no significant differences in median or quartile values. However, session time data reveals more variability:

- The PQC time box plot is relatively compact, suggesting that all tested PQC algorithm combinations achieve similar handshake latencies.
- The Hybrid time box plot is more dispersed, indicating that algorithm choice has a larger influence on latency in Hybrid configurations.

Outlier detection using the interquartile range (IQR) method identified *Sphincs+-192f* as a high-latency outlier. Given its deviation from the main distribution in both PQC and Hybrid cases, this parameter set is excluded from subsequent prototype candidate analysis to maintain a focus on representative, performant configurations.

5.2 SCP: Aggregate Efficiency

Following the handshake performance evaluation (Subsection 5.1), we extended the analysis to full SSH SCP file transfers to assess how key exchange and signature algorithm choices influence the established end-to-end session efficiency. As SCP operates over the TCP transport protocol, additional TCP headers and acknowledgement packets inherently generate extra traffic, increasing the overall size of file transfer sessions. This overhead was quantified by subtracting both the initial SSH session establishment size and the raw file size from the total file transfer session size.

We first examined the proportional relationship between file transfer session size and completion time to identify any irregularities or performance anomalies. Across all tested algorithms, no malfunctions or unexpected outliers were observed; all results fell within expected variance bounds, indicating stable and predictable performance for every implemented variant.

Figure 4 compares average SCP time and size across signature algorithms for payloads of 1 MB, 10 MB, 100 MB and 200 MB. At larger file sizes, the choice of cryptographic algorithm has less impact on transfer time; however, Falcon tends to be the most size-efficient, while Dilithium is marginally faster in Hybrid mode.

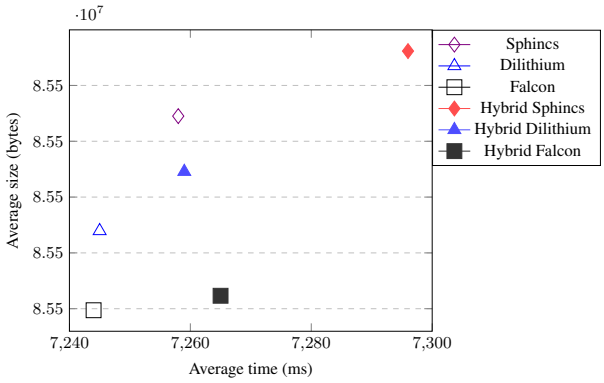

Fig. 4: Average SCP time vs. size across 1 MB, 10 MB, 100 MB and 200 MB

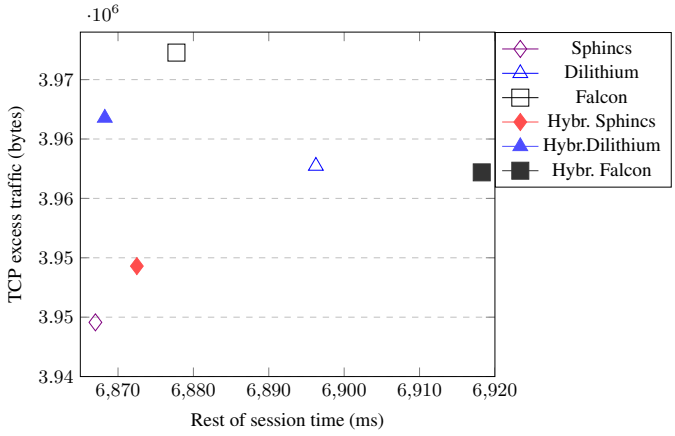

Fig. 5: Residual SCP session metrics: TCP overhead traffic (session size minus SSH session size and file size) vs. residual transfer time (excluding handshake).

Figure 5 presents the averaged residual session metrics, where the Y-axis represents TCP overhead (in bytes) and the X-axis shows the residual transfer time (excluding SSH session establishment). This residual phase isolates the sustained data transfer component, independent of handshake costs.

Across both PQC and its hybrid implementations, *Sphincs* consistently exhibits the smallest residual size and shortest residual duration. However, this apparent efficiency is misleading: *Sphincs* remains the slowest overall and produces the largest total session size due to its exceptionally large signatures.

In PQC-only mode, *Dilithium* and *Falcon* differ in transfer time by only ~ 1 ms, yet *Falcon* achieves smaller total session sizes. In Hybrid configurations, *Dilithium* is marginally faster, while *Falcon* maintains superior size efficiency.

5.3 KEM analysis: Kyber vs Kyber-90s in SSH Handshakes

Kyber has two variants of the main pseudorandom function (PRF) options and other symmetric sub-primitives:

- Standard Kyber $\rightarrow$ all symmetric sub-primitives are instantiated with Keccak-based FIPS 202 functions (XOF = SHAKE-128 (for generating the public matrix), PRF / KDF = SHAKE-256, H = SHA3-256, G = SHA3-512). This design ties Kyber tightly to the SHA-3 family.
- Kyber-90s $\rightarrow$ PRF based on AES-256/SHA-2 in counter mode (AES-CTR), designed for faster performance when hardware AES acceleration (AES-NI) is available. The XOF is replaced with AES-256 in CTR mode, the PRF is replaced with AES-256 in CTR mode keyed differently, H and G are instantiated with SHA-2 functions (SHA-256 and SHA-512) instead of SHA-3 (Avanzi et al. (2021)).

NIST excluded Kyber-90s from the ML-KEM standard (National Institute of Standards and Technology, 2022a) to simplify adoption by selecting a single, SHAKE/SHA-3 based variant, though AES-based designs may still be relevant in niche or constrained deployments.

To isolate the impact of the KEM choice from the large performance variance introduced by different signature schemes (e.g., compact Falcon vs large SPHINCS+), we computed the *average* SSH handshake session size and latency for each Kyber parameter set, aggregated across all tested PQC signatures. These results, shown in Table 11, focus solely on the contribution of the KEM implementation to overall performance and Security Levels.

Observed trends. The aggregated results confirm that **handshake session size is virtually identical** between Kyber and Kyber-90s at the same security level, size differences are minimal (<1%), meaning storage and transmission overhead between the two variants is practically the same.

Latency favors Kyber-90s at Level 1, but reverses at Levels 3 and 5. Since both variants share identical public key and ciphertext lengths, differing only in their internal pseudorandom function (SHAKE vs AES-256-CTR). Kyber-90s does not outperform Kyber at higher security levels in latency and shows measurable variation:

- **Level 1 (Kyber512):** Kyber-90s is $\approx 4.4\%$ faster.

Table 11: Average SSH handshake size and time for Kyber vs Kyber-90s across all tested signatures.

KEM Parameter Set	Average Size (bytes)		Average Time (ms)		NIST Security
	Kyber	Kyber-90s	Kyber	Kyber-90s	
Kyber512	61,792	61,765	390.9	373.6	Level 1
Kyber768	62,460	62,875	390.7	413.4	Level 3
Kyber1024	63,438	63,954	384.7	422.2	Level 5

– **Level 3 (Kyber768):** Kyber-90s is $\approx 5.8\%$ slower.
– **Level 5 (Kyber1024):** Kyber-90s incurs $\approx 9.7\%$ slowdown.

Benchmarks show mixed results — sometimes Kyber is faster at larger parameter sets, sometimes Kyber-90s wins, depending on whether SHA-2/AES acceleration is present and on implementation. There are published implementations of Kyber-90s optimized for constrained devices (ESP32) demonstrating that 90s can be very competitive at L1 on some platforms (Segatz and Hafiz (2025)). The observed Kyber-90s are faster than Kyber, which is plausibly caused by the presence of CPU instructions that accelerate AES. Kyber-90s does not outperform Kyber at higher security levels, suggesting that larger key and ciphertext sizes negate AES performance advantages — consistent with the Kyber designers' assertion (Avanzi et al. (2019)) that the AES-based variant is only preferable when AES is accelerated in hardware.

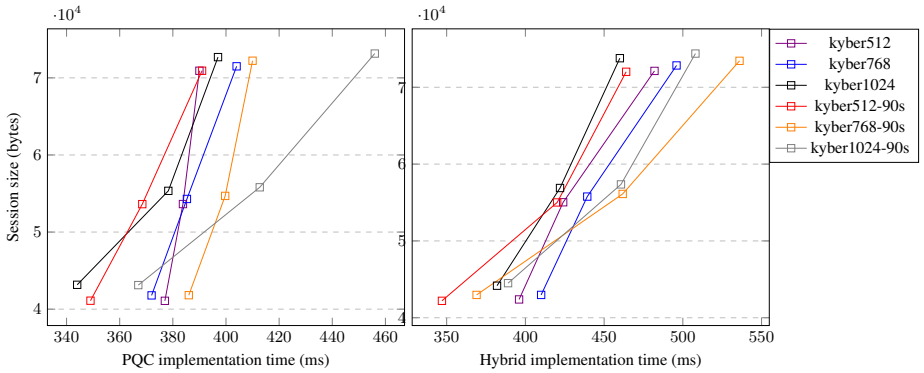

Fig. 6: Kyber parameters of SSH session time and size: PQC vs Hybrid

Figure 6 presents the minimum, average, and maximum data for both time and size, distinguishing between PQC implementations on the left and their hybrid implementations on the right of every Kyber parameter set. To sum up, Kyber's contribution to session size and time remained consistent, exhibiting linear scaling with security level and showing low variance across signature pairings. When combined with classical ECC in hybrid handshakes, Kyber added only a minimal size overhead compared to PQC mode.

5.4 Digital Signature Algorithms Analysis

To complement the key encapsulation mechanism results, we analyzed the performance characteristics of the PQC digital signature algorithms (Falcon, Dilithium, Sphincs+) used in our SSH handshake experiments. The goal was to quantify their impact on overall handshake efficiency and to identify trade-offs relevant to post-quantum migration strategies.

Table 12 summarizes the average handshake size and time for each signature algorithm, computed across all Kyber parameter sets. This averaging smooths out the effect of KEM size scaling and isolates the performance contribution of the signature component.

Observed Trends. From the aggregated results, several clear patterns emerge:

- **Handshake size is dominated by signature size.** Sphincs+ variants have the largest signatures (17 KB for -128f, 36 KB for -192f), making them significantly larger than lattice-based schemes. Falcon and Dilithium are much more practical for SSH, with moderate signature sizes (0.7– 46 KB) and handshake times (<500 ms for hybrid and <400 ms PQC).
- **Handshake time differences are smaller than size differences.** While Falcon consistently shows good size efficiency, its handshake times are not significantly faster than Dilithium, and the highest security Level5 Dilithium5-AES slightly outperforms Falcon1024 in speed (2 ms).
- **Hybrid vs PQC impact is modest.** In PQC, introducing a classical ECC component into the handshake increases average handshake size by ≈1.4–3.5% depending on algorithm, so size overhead for hybrid handshakes is relatively small. Latency impact is more pronounced than size overhead for hybrid PQC handshakes:
  - Falcon: + 15–43 ms (3.9 – 10.9 %)
  - Dilithium: + 12–91 ms (3.1 – 32.7 %)
  - Sphincs+: + 39–76 ms (9.8 – 17.2 %)
- **Sphincs+ paradox.** Although Sphincs+ shows competitive residual (post-handshake) SCP performance in Section 5.2, its handshake cost is the highest in both size and time, making it less suitable for frequent connection establishments in bandwidth-limited environments.

Figure 7 illustrates the performance data of all tested digital signature algorithms in terms of size and time. The left side displays the results from the PQC implementation and Classical ECC, while the right side highlights the results from the Hybrid implementation. From the interpretation of Figure 7, the fastest time results are located further left in the plot.

Security-Level Perspective. Falcon-512 and Dilithium2-AES offer NIST Level 1 security, while Falcon-1024 and Dilithium5-AES reach Level 5. Dilithium3 is Level 3. The higher-security variants predictably increase handshake sizes and times, but the growth is far more pronounced for Sphincs+, whose large signature sizes scale sharply with security level.

Table 12: Average SSH handshake size and time for post-quantum signature algorithms (averaged over all Kyber variants).

Signature Algorithm	**Average Size (bytes)**		**Average Time (ms)**	
	PQC	Hybrid	PQC	Hybrid
Falcon-512	42,009	43,201	384	399
Falcon-1024	49,169	50,864	395	438
Dilithium2-AES	48,438	49,838	387	399
Dilithium3	54,590	56,002	370	491
Dilithium5-AES	61,284	63,352	393	461.5
Sphincs+-128f	71,903	73,009	400	439
Sphincs+-192f	111,604	113,146	443	519

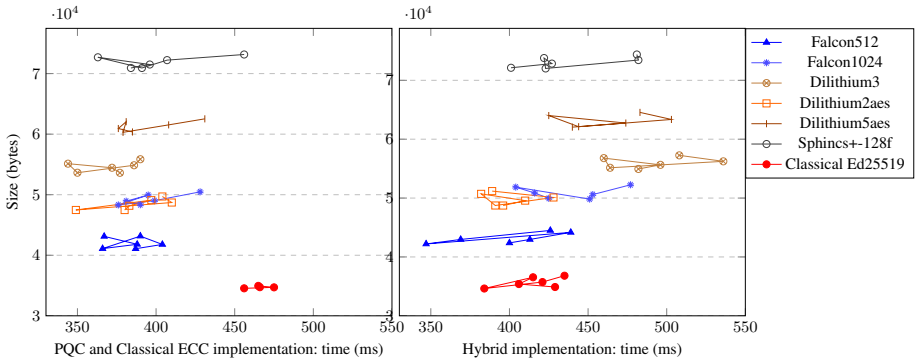

Fig. 7: Digital signature algorithms parameters of SSH session time and size: PQC vs Classical ECC vs Hybrid

Practical Takeaways. From an operational deployment perspective:

- *Falcon* is the best candidate for minimizing handshake size, which is valuable for low-bandwidth or high-latency links.
- *Dilithium* variants offer competitive speed and simpler implementation properties, making them suitable when size is less critical.
- *Sphincs+* should be reserved for scenarios prioritizing conservative, hash-based security over performance, given its handshake overhead.

These results highlight that, in the context of SSH, the choice of signature algorithm can have a larger impact on handshake size than the choice between Kyber variants for KEM, emphasizing the need to consider signature size in PQC migration strategies.

5.5 NIST Security Levels Relevance to PQC Implementation

National Institute of Standards and Technology (2016) defined *security levels* for PQC algorithms as part of its standardization process. Secure communication protocols such as SSH typically operate at the equivalent of Level 1. For example, RSA-2048 or ECDSA P-256 aligns with AES-128 security and is sufficient for most enterprise and short-to-medium data retention uses (IETF PQC Working Group, 2024). Levels 3–5, by contrast, offer higher security margins suitable for long-term protection of sensitive or archival data.

This study evaluated the performance of PQC algorithms and provides recommendations for PQC migration covering security levels:

- **Kyber512 combination with Falcon512 or Dilithium2-AES:** Level 1 — fastest performance and smallest sizes; adequate for most enterprise SSH deployments with typical data retention requirements.
- **Kyber768 combination with Dilithium3:** Level 3 — stronger security at a moderate performance cost.
- **Kyber1024 combination with Dilithium5-AES or Falcon1024:** Level 5 — maximum standardized security strength; introduces significant size overhead with only marginal handshake time impact.

From an operational perspective, selecting the security level for SSH should balance performance impact against the required security margin. For short-lived SSH sessions, lower levels can maximize throughput, whereas archival or compliance-driven environments may justify the higher costs of Level 3 or above.

6 Conclusions and Future Work

This study conducted a comprehensive empirical evaluation of post-quantum cryptographic algorithms in their pure form and in hybrid combinations with classical Elliptic Curve Cryptography within SSH protocol handshakes and file transfer sessions. By systematically measuring handshake latency, session size, and sustained transfer performance across multiple NIST security levels, we established a **dataset linking cryptographic parameter choices** to real-world operational outcomes.

These baselines and PQC variants allow us to directly observe how key length, ciphertext size, and signature size translate into measurable SSH handshake performance. By aligning each algorithm with its NIST security level, it is possible to quantify how the required security margin affects real-world SSH performance. This provides an evidence-based **justification for algorithm selection in post-quantum migration strategies**, ensuring that the security gain is achieved by selecting the most efficient combination of algorithms.

The results show that the PQC digital signature algorithm signature size is relevant to SSH session size; a correlation is observed between SSH sessions and signature size. No significant correlation was observed between the PQC algorithm's key sizes and SSH session time performance:

- **Falcon** consistently achieves the smallest handshake sizes and remains competitive in latency, making it suitable for bandwidth-sensitive environments.
- **Dilithium** offers stable performance and strong standardization support, with marginal speed advantages in Hybrid configurations.
- **Sphincs+**, while providing conservative hash-based security, imposes significant size and latency costs and is best reserved for scenarios prioritizing long-term cryptographic resilience over efficiency.
- **Kyber** exhibits consistently low variance in handshake times across NIST levels, making it predictable and practical for deployment planning.
- Hybrid modes preserved classical trust anchors at only modest performance cost, making them practical for transitional deployments.

From an operational perspective, the choice of *security level* and *signature scheme* has a more substantial performance impact. Table 13 summarizes recommended PQC algorithm pairings for SSH migration. Overall, our analysis suggests that Falcon paired

Table 13: Recommended PQC algorithm choices for SSH migration

Scenario	**Recommendation**	**Rationale**
Bandwidth-limited	Falcon-512 + Kyber512	Level 1 security; Smallest handshake
Long-term security	Falcon-1024 + Kyber1024	Level 5; Balanced speed and size
Balanced migration	Dilithium3 + Kyber1024	Level 3 & 5; Best speed / size trade-off
High-trust Hybrid	ecdh-nistp512-Kyber1024 wtih Ed25519	Level 5 PQC + Classical trust anchor
Conservative assurance	Sphincs+-128f/192f + Kyber768	Hash-based; Highest performance cost

with Kyber provides favorable trade-offs between size and performance. However, the adoption of Falcon requires careful consideration of side-channel resistance and implementation robustness, given its reliance on Gaussian sampling. Importantly, the experimental results indicate that **the most effective combination is Dilithium3 with Kyber1024**. While this pairing does not achieve the smallest session size, it delivers the shortest session time and ensures a balanced security profile by combining NIST levels 3 and 5.

Future research should extend these experiments beyond controlled laboratory conditions to more realistic settings. Evaluating PQC-enabled SSH in wide-area and high-latency networks would clarify how packet loss and congestion influence handshake and session overheads. Multi-client scalability and server-side load analysis are also critical for assessing performance in enterprise and cloud deployments. Finally, future work should examine the integration of PQC into public key infrastructures, including hybrid certificates, and assess compliance implications under regulatory frameworks such as the European Union NIS2 and the Cyber Resilience Act.

References

Alnahawi, N., Wiesmaier, A., Grasmeyer, T., Geißler, J., Zeier, A., Bauspieß, P., Heinemann, A. (2021). On the state of post-quantum cryptography migration, *INFORMATIK 2021*, Gesellschaft für Informatik, Bonn, pp. 907–941.

Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Stehlé, D., Schwabe, P. et al. (2021). CRYSTALS-Kyber algorithm specification (round 3), *Technical report*, PQCRystals. `https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf`.

Avanzi, R. et al. (2019). Crystals-kyber round 2 specification, `https://pq-crystals.org/kyber/data/kyber-specification-round2.pdf`. Accessed: 2025-08-09.

Barker, E., Dang, Q., Roginsky, A. (2015). Recommendation for key management: Application-specific key management guidance, *NIST Special Publication 800-57 Part 3 Revision 1*, National Institute of Standards and Technology.
`https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt3r1.pdf`

Barker, W., Polk, W., Souppaya, M. (2021). Getting ready for post-quantum cryptography: Exploring challenges associated with adopting and using post-quantum cryptographic algorithms. Last accessed: 2024-10-13, `https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04282021.pdf`.

Bernstein, D. J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y. (2012). High-speed high-security signatures, *Cryptographic Hardware and Embedded Systems – CHES 2011*, Vol. 6917 of *Lecture Notes in Computer Science*, Springer, pp. 124–142.
`https://ed25519.cr.yp.to/ed25519-20110926.pdf`

Bindel, N., Brendel, J., Fischlin, M., Goncalves, B., Stebila, D. (2019). Hybrid key encapsulation mechanisms and authenticated key exchange, *in* Ding, J., Steinwandt, R. (eds), *Proc. 10th International Conference on Post-Quantum Cryptography (PQCrypto) 2019*, Vol. 11505 of *LNCS*, Springer, pp. 202–226.

Crockett, E., Paquin, C., Stebila, D. (2019). Prototyping post-quantum and hybrid key exchange and authentication in tls and ssh, *Cryptology ePrint Archive* . Access online: `https://ia.cr/2019/858`.

IETF PQC Working Group (2024). Post-quantum cryptography for engineers, IETF Draft (draft-ietf-pquip-pqc-engineers-06), Section 11. Accessed: 2025-08-10.

IETF SSH Working Group (2009). Elliptic curve algorithm integration in the secure shell (ssh) transport layer, RFC 5656.
`https://datatracker.ietf.org/doc/html/rfc5656`

Jacob W. S. Schneider, P. S. (2024). NIST Releases Three Post-Quantum Cryptography Standards | Insights | Holland & Knight — hklaw.com, `https://www.hklaw.com/en/insights/publications/2024/08/nist-releases-three-post-quantum-cryptography-standards`. [Accessed 13-10-2024].

Joseph, D., Misoczki, R., Manzano, M. (2024). Transitioning organizations to post-quantum cryptography, *Nature* **605**, 237–243.

Kampanakis, P., Stebila, D., Friedl, M., Hansen, T., Sikeridis, D. (2020). Post-quantum public key algorithms for the secure shell (ssh) protocol, IETF Internet-Draft, draft-kampanakis-curdle-pq-ssh-00. Available at IETF Datatracker.
https://datatracker.ietf.org/doc/html/draft-kampanakis-curdle-pq-ssh-00

Kreutzer, M., Niederhagen, R., Waidner, M., Gespräch, E. (2018). Next generation crypto. Last accessed: 2024-04-14, https://www.sit.fraunhofer.de/fileadmin/dokumente/studien_und_technical_reports/EberbacherBroschuere_prefinal_V10.pdf?_=1520946028.

Mosca, M. (2018). Cybersecurity in an era with quantum computers: Will we be ready?, *IEEE Security Privacy* **16**(5), 38–41.

National Institute of Standards and Technology (2016). Security (evaluation criteria), https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria). Accessed: 2025-08-10.

National Institute of Standards and Technology (2022a). Official comment on the selection of crystals-kyber as the nist post-quantum cryptography standard, https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/selected-algos-2022/official-comments/crystals-kyber-selected-algo-official-comment.pdf. Accessed: 2025-08-09.

National Institute of Standards and Technology (2022b). Official comment on the selection of falcon as a nist post-quantum cryptography candidate, NIST PQC Standardization Project. Accessed: 2025-09-07.
https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/selected-algos-2022/official-comments/falcon-selected-algo-official-comment.pdf

National Institute of Standards and Technology (2023). Fips pub 186-5: Digital signature standard (dss), https://doi.org/10.6028/NIST.FIPS.186-5. Accessed: 2025-08-09.

National Institute of Standards and Technology (2024a). Announcing issuance of federal information processing standards (fips) 203, module-lattice-based key-encapsulation mechanism standard; fips 204, module-lattice-based digital signature standard; and fips 205, stateless hash-based digital signature standard, *Federal Register* **89**(157), 66052–66057.

National Institute of Standards and Technology (2024b). FIPS 203: ML-KEM – module lattice-based key encapsulation mechanism, *Federal Information Processing Standards Publication 203*, U.S. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD.
https://doi.org/10.6028/NIST.FIPS.203

National Institute of Standards and Technology (2024c). FIPS 204: ML-DSA – module lattice-based digital signature algorithm, *Federal Information Processing Standards Publication 204*, U.S. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD.
https://doi.org/10.6028/NIST.FIPS.204

National Institute of Standards and Technology (2024d). FIPS 205: SLH-DSA – stateless hash-based digital signature algorithm, *Federal Information Processing Standards Publication 205*, U.S. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD.
https://doi.org/10.6028/NIST.FIPS.205

National Institute of Standards and Technology (2024e). Post-quantum cryptography: Publications, https://csrc.nist.gov/Projects/post-quantum-cryptography/publications. Accessed July 26, 2025.

Nina, B., Johannes, B., Luca, G., Tobias, S., Johannes, W. (2019). X.509-compliant hybrid certificates for the post-quantum transition, *Journal of Open Source Software* **4**(40), 1606. Last accessed: 2024-04-14, https://doi.org/10.21105/joss.01606.

Ott, D., Peikert, C., participants, o. w. (2019). Identifying research challenges in post quantum cryptography migration and cryptographic agility. Last accessed: 2024-04-23, https://arxiv.org/abs/1909.07353.
https://arxiv.org/abs/1909.07353

Ounsworth, M., Pala, M. (2024). Composite signatures for use in internet pki, ietf, 8 june 2024, during conference PQCrypto2021. Last accessed: 2024-04-14, https://www.ietf.org/id/draft-ounsworth-pq-composite-sigs-07.html.

Paquin, C., Stebila, D., Tamvada, G. (2019). Benchmarking post-quantum cryptography in tls, *Technical Report 2019/1447*, IACR Cryptology ePrint Archive.
https://eprint.iacr.org/2019/1447

Pierre-Alain, F., Jeffrey, H., Paul, K., Vadim, L., Thomas, P., Thomas, P., Thomas, R., S., W. G., Whyte, Z. Z. (2024). Falcon: Fast-fourier lattice-based compact signatures over ntru, https://www.di.ens.fr/~prest/Publications/falcon.pdf. [Accessed 12-09-2025].

Prest, T., Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Stehlé, D., Zaverucha, G. (2020). Falcon: Fast-fourier lattice-based compact signatures over ntru, https://falcon-sign.info. Submission to NIST PQC Standardization Project, Round 3.

Rossi, M. (2021). Pqc transition - anssi views, during conference PQCrypto2021. Last accessed: 2024-04-14, http://pqcrypto2021.kr/download/program/PQC_transition_in_France.pdf.

Schwabe, P., Stebila, D., Wiggers, T. (2020). Post-quantum tls without handshake signatures, *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS '20)*, ACM, pp. 1234–1243.

Segatz, F., Hafiz, M. I. A. (2025). Efficient implementation of crystals-kyber key encapsulation mechanism on esp32, *arXiv preprint arXiv:2503.10207v1* . https://arxiv.org/abs/2503.10207v1.

Sikeridis, D., Kampanakis, P., Devetsikiotis, M. (2020a). Assessing the overhead of post-quantum cryptography in tls 1.3 and ssh, *Proceedings of the 16th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '20, Association for Computing Machinery, New York, NY, USA, p. 149–156.
https://doi.org/10.1145/3386367.3431305

Sikeridis, D., Kampanakis, P., Devetsikiotis, M. (2020b). Assessing the overhead of post-quantum cryptography in tls 1.3 and ssh, *Proceedings of CoNEXT '20*, Barcelona, Spain, pp. 1–7.

Stebila, D., Fluhrer, S., Gueron, S. (2021). Hybrid key exchange in tls 1.3, IETF Internet-Draft, draft-ietf-tls-hybrid-design-03. Last accessed: 2025-09-07.
https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-03

Tran, D. D., Ogata, K., Escobar, S., Akleylek, S., Otmani, A. (2024). Formal analysis of post-quantum hybrid key exchange ssh transport layer protocol, *IEEE Access* **12**, 1672–1687.

Received February 13, 2025 , revised September 28, 2025, accepted November 5, 2025