J. A. Montenegro, R. Rios and J. Lopez-Cerezo, "A Performance Evaluation Framework for Post-Quantum TLS", *Future Generation Computer Systems*, In Press.
NICS Lab. Publications: https://www.nics.uma.es/publications

A Performance Evaluation Framework for Post-Quantum TLS

José A. Montenegro[a,1], Ruben Rios[a] and Javier López-Cerezo[a]

[a]*Network, Information and Computer Security (NICS) lab, Universidad de Malaga, Malaga, 29071, Spain*

ARTICLE INFO

Keywords:
Post-Quantum Cryptography (PQC)
Transport Layer Security (TLS)
NIST Standards
Handshake Performance
Quantum-Resistant Protocols

ABSTRACT

Quantum computers pose a significant threat to widely used cryptographic schemes, making it crucial to urgently shift to post-quantum protocols that ensure the long-term security of communications. This paper presents a framework for facilitating the transition to a quantum-resistant web by enabling the evaluation of the impact of integrating post-quantum and hybrid cryptographic primitives in the TLS protocol. By leveraging the OpenSSL library, the framework enables the seamless evaluation and comparison of key encapsulation mechanisms (KEMs) and signature schemes provided by the Open Quantum Safe project. The proposed framework is used to analyze different TLS configurations involving classic, hybrid and post-quantum primitives, with a focus on those selected by NIST. The study concentrates on evaluating the performance impact and the data overhead introduced by configurations involving primitives at different security levels. It starts by benchmarking standalone cryptographic primitives and continues with the evaluation of TLS configurations with hybrid and post-quantum KEMs under controlled network conditions, as this is the first natural step for a transition to a quantum-safe TLS. An impact analysis on real-world scenarios follows. Finally, our evaluation concentrates on providing insight into the cost of shifting to TLS configurations involving only hybrid or post-quantum primitives.

1. Introduction

The Internet stands as one of the greatest technological advances in human history. Beyond connecting devices, it has profoundly transformed the way people communicate and conduct everyday activities, such as business and entertainment. This revolution has impacted virtually every aspect of modern life and continues to be a key driver of technological and social innovation.

The development and mass adoption of the Internet as we know it today, would have not been possible without existing security protocols, such as the Transport Layer Security (TLS) protocol [1]. These security protocols rely on symmetric and asymmetric cryptographic primitives to ensure the confidentiality, integrity and authenticity of communications. They provide the necessary trust to conduct financial transactions, and share sensitive information among individuals and businesses. However, recent advancements in quantum computing are shaking the trust foundations of the Internet.

The majority of modern cryptographic algorithms used to secure communications rely on complex mathematical problems that are exceptionally challenging for current computers to solve. For instance, the security of the RSA algorithm is grounded in the difficulty of factorizing large integers. Similarly, other algorithms, such as the Diffie-Hellman key exchange, are based on the infeasibility of calculating discrete logarithms. Unfortunately, these problems could be solved by quantum computers exponentially faster than classical computers [2].

Despite the fact that a cryptographically relevant quantum computers (CRQC)[1] does not yet exist, there is a looming threat that encrypted data could be stored now, with the intent to decrypt it once quantum computers become available [3]. This underscores the importance of making a swift transition to cryptographic algorithms that can withstand quantum attacks. Such algorithms already exist and they are referred to as quantum-resistant or post-quantum cryptography (PQC) algorithms.

After six years of a public competition organized by the U.S. National Institute of Standards and Technology (NIST), three algorithms were published as FIPS (Federal Information Processing Standard) in the summer of 2024 [4]. The focus now shifts to the industry, which bears the responsibility of integrating these post-quantum algorithms into security protocols to safeguard contemporary communications against the potential risks posed by future quantum computers.

The Internet Engineering Task Force (IETF) is actively working to standardize the integration of PQC primitives into communication protocols, such as TLS [5, 6]. These efforts have prompted major industry players to begin incorporating PQC solutions into their systems [7, 8, 9]. However, despite these initiatives, PQC algorithms have yet to achieve widespread adoption within TLS, and early integration attempts have faced notable challenges [10].

The primary contribution of this paper is a framework that enables seamless integration and comparison of traditional, hybrid, and post-quantum cryptographic primitives within the TLS protocol. The framework focuses on evaluating the impact of integrating these primitives on handshake performance and the volume of data exchanged under both

*Corresponding author
jmmontes@uma.es (J.A. Montenegro); ruben.rdp@uma.es (R. Rios); fjlc@uma.es (J. López-Cerezo)
ORCID(s): 0000-0001-6967-0801 (J.A. Montenegro); 0000-0002-6251-4897 (R. Rios); 0009-0004-7531-0377 (J. López-Cerezo)

[1] NSA defines it as a quantum computer capable of breaking a real-world cryptographic system that would be impossible to attack with a traditional computer.

controlled and realistic network conditions, with configurable packet loss and delay. The main motivation behind the development of this framework is to facilitate an informed transition to a quantum-resistant web.

The rest of this paper is organized as follows. In Sec. 2, prior research and developments in post-quantum cryptography and its integration into the TLS protocol are reviewed. Sec. 3 provides an overview of the TLS protocol focusing on the incorporation of post-quantum algorithms into the handshake process, along with the mathematical foundations of the post-quantum primitives used. The evaluation framework is introduced in Sec. 4. Sec. 5 evaluates the cryptographic primitives used for KEM and signature. Next, Sec. 6 focuses on assessing the current status of the TLS protocol with post-quantum and hybrid algorithms in KEM. Sec. 7 addresses challenges in deploying post-quantum TLS, such as handling mis-aligned configuration, ensuring mutual authentication, and simulating real-word network conditions. The future of post-quantum TLS is examined in Sec. 8, emphasizing the need to adopt hybrid and fully post-quantum solutions, considering standardized signature algorithms and novel proposals. Sec. 9 presents the conclusions of this work and suggests lines of future work.

2. Related Work

Post-quantum cryptography has been extensively studied across various application domains in recent years. In addition to the integration of post-quantum cryptographic primitives into the TLS protocol, two areas that have received particular attention are the Internet of Things (IoT) and blockchain technology. In the context of IoT, research has primarily focused on the cost and feasibility of implementing post-quantum primitives on resource-constrained devices [11, 12]. In the blockchain domain, post-quantum cryptography has been applied to various challenges, including enhancing consensus mechanisms and creating quantum-resistant transactions [13, 14].

The preliminary experiments integrating post-quantum cryptographic primitives into the TLS protocol were conducted by Google [15, 16], and subsequently expanded through collaborative efforts with other entities, including Cloudflare [17, 18]. These studies concentrated on evaluating the performance of real-world connections employing hybrid post-quantum key exchange. Client browsers and servers were modified to support selected hybrid key exchange schemes in TLS 1.3. While these experiments provide highly realistic insights, their replication is challenging due to the required infrastructure.

A substantial number of research studies have evaluated the behavior of post-quantum TLS, primarily TLS 1.2, in Internet of Things (IoT) environments. The MbedTLS library has been used as a key research tool in numerous studies, including [19], [20], [21], [22] and [23]. The findings of these studies emphasize the need for optimized solutions that strike a balance between security, performance and energy efficiency. However, it should be noted that the aforementioned studies relied on post-quantum algorithms that were available at that time, integrating them through custom implementations that were tailored to specific devices. Consequently, it is important to note that the conclusions drawn cannot be automatically extrapolated to current or future post-quantum schemes, as it is not straightforward to replace the evaluated algorithms with newer versions.

Another relevant line of research addresses the challenges introduced by post-quantum signatures, including performance concerns and the increased transmission overhead caused by large key and certificate sizes. To mitigate these issues, a solution known as KEMTLS [24] has been proposed. This solution uses a novel KEMs not only for key exchange but also for server authentication, thereby alleviating the overhead imposed by post-quantum signatures. The authors highlight the substantial improvements of this proposal compared to the standard approach. However, it is worth noting that this proposal has not yet been standardized [25], which raises compatibility concerns with the TLS standard.

Our work builds upon the research conducted by the Open Quantum Safe (OQS) community [26]. Previous studies, have also been used their work as a basis. For example, [27] developed a client-side handshake implementation using OQS-OpenSSL 1.1.1 and a Nginx server to evaluate performance. This study simulates real network conditions and assesses KEMs and digital signature algorithms available at that time. However, their evaluation is limited as it considers only a single key size, without accounting for the performance impact introduced by using different key sizes. In a recent study [28], the author utilizes OQS and the OpenSSL `s_time` tool, to measure the number of connections per second. However, the measurements used CPU time rather than real-time, which can be misleading as it deviates from real-world results. Moreover, the measurements were conducted while keeping the KEM or signature algorithm in the pre-quantum state. Specifically, when benchmarking the key exchange, the algorithm used was X25519, while the signature algorithm was ECDSA with P-256. Consequently, the measurements were not performed with both options using post-quantum algorithms.

3. Background

This section provides the necessary background for the rest of the paper. First, it presents an overview of the TLS protocol, with a focus on the cryptographic operations and the messages exchanged during the handshake phase. Next, it outlines the foundations of the post-quantum algorithms that will enable a quantum-safe TLS.

3.1. TLS Protocol Overview

Transport Layer Security (TLS) is a critical security protocol used across various Internet applications, including web browsing, email, file transfers, and messaging apps. Before any data transmission occurs, TLS establishes a secure

connection to prevent third parties from intercepting, altering, or forging messages, thereby ensuring the authenticity, confidentiality, and integrity of online communications. The latest version, TLS 1.3, is specified in RFC 8446, while TLS 1.2 remains widely used. Although deprecated, some systems continue to rely on earlier TLS versions.

The TLS protocol consists of two phases. During the initial handshake phase, the client and server negotiate the cryptographic algorithms and key material required to secure the subsequent data transmission phase. Research efforts to adopt post-quantum cryptography focus on the handshake phase, as it is the most vulnerable to quantum attacks due to its reliance on asymmetric cryptography.

Figure 1 illustrates both the messages exchanged and the cryptographic operations associated with them during the normal operation of the protocol. The cryptographic operations used for key agreement are known as key encapsulation mechanisms (KEMs), and those used for authentication involve the standard public key signatures and verifications. While the latter are well understood, it is useful to review the three core algorithms that comprise KEMs:

- **KeyGen**() $\rightarrow (pk, sk)$: A probabilistic key generation algorithm, which generates a public key pk and a secret key sk.
- **Encaps**(pk) $\rightarrow (ct, ss)$: A probabilistic encapsulation algorithm, which takes as input a public key pk and outputs a ciphertext ct and shared secret ss.
- **Decaps**(sk, ct) $\rightarrow ss$: A decapsulation algorithm, which takes as input a secret key sk and ciphertext ct and outputs a shared secret ss, or in some cases, a distinguished error value.

To further elaborate, the handshake phase is organized into three distinct stages. In the *key exchange phase*, the `ClientHello` message contains a list of supported algorithms in descending order of client preference. To enhance this process and facilitate the integration of post-quantum algorithms into TLS, the Internet Draft [29] introduces a hybrid key exchange algorithm to the standard. This approach allows to achieve post-quantum security while preserving the security guarantees provided by traditional algorithms.

For the client's share, the key_exchange field includes the concatenation of the public key outputs from the corresponding KEMs' `KeyGen` operations for KEM algorithms, and the (EC)DH ephemeral key share for (EC)DH groups. The elliptic curves used in TLS 1.3 are secp256r1, secp384r1, and secp521r1 (also known as NIST P-256, P-384, and P-521), as well as x25519 and x448. For KEM-based key exchange, ML-KEM with three different key lengths is used.

In contrast, the `ServerHello` message contains only the key share corresponding to the algorithm selected by the server from those offered by the client. Therefore, the key_exchange field includes the concatenation of the ct outputs from the corresponding KEMs' `Encaps` algorithms for KEM algorithms, and the (EC)DH ephemeral key share for (EC)DH groups. Once the client receive the `ServerHello` message, the client executes the corresponding `Decaps()` to derive the shared secret (ss), ensuring that both parties obtain the same value.

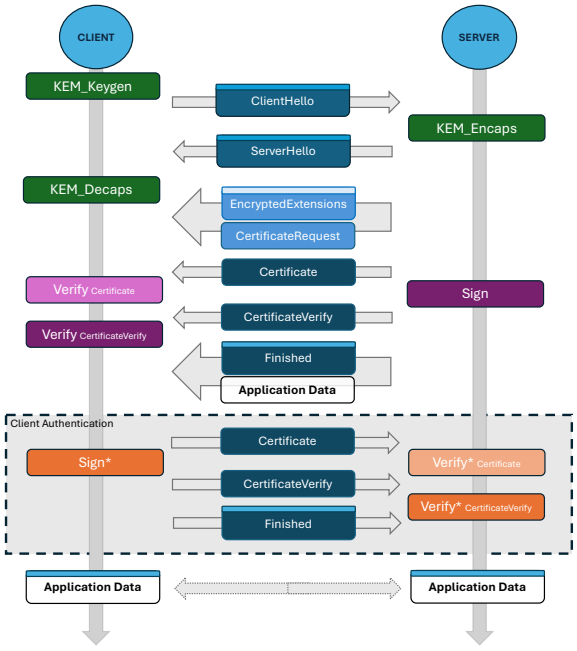


Figure 1: TLS 1.3 Message Flow (based on RFC 8446 and draft-ietf-tls-hybrid-design-12)

The *server parameters phase* comprises only two messages, `EncryptedExtensions` and `CertificateRequest`, with the latter being sent only in case the server request the client to authenticate. In its simplest form, only the server is authenticated to the client (server-only or server-side), which is common in web scenarios. However, in some cases, mutual authentication is recommended.

During the *authentication phase*, three types of messages are exchanged. The `Certificate` message typically contains the certificate of the endpoint. The certificate is signed by a trusted authority. This signature must be verified using the signature algorithm specified in the certificate. The `CertificateVerify` message contains a signature over the entirety of the handshake. The server signs the information using the private key associated with the certificate sent in the previous message. Upon receiving the message, the client is responsible for verifying the signature. Traditional signature algorithms include RSA, the Elliptic Curve Digital Signature Algorithm (ECDSA), and the Edwards-Curve Digital Signature Algorithm (EdDSA). The elliptic curves used are the same as those employed during the key exchange phase. In contrast, the available post-quantum (PQ) signature algorithms are listed in Table 1. The `Finished` message confirms the integrity of the handshake and the establishment of the shared secret. After its exchange, both the client and the server derive the keying material necessary for encrypting application-level data using authenticated encryption.

In the case of mutual authentication, the client must send the `Certificate`, `CertificateVerify`, and `Finished` messages. The transmission of these messages requires the execution of

Table 1
PQC schemes evaluated and their security levels

		NIST Security Category		
	Algorithm name	**Level I**	**Level III**	**Level V**
PKE/KEM	CRYSTALS-KYBER / ML-KEM	KYBER512 / ML-KEM512	KYBER768 / ML-KEM768	KYBER1024 / ML-KEM1024
Signature	CRYSTALS-DILITHIUM / ML-DSA	–	DILITHIUM3 / ML-DSA65	DILITHIUM5 / ML-DSA87
	FALCON	FALCON512	–	FALCON1024
	SPHINCS$^+$ / SLH-DSA	SPHINCS$^+$-128S / SLH-DSA-128S	SPHINCS$^+$-192S / SLH-DSA-192S	SPHINCS$^+$-256S / SLH-DSA-256S
		SPHINCS$^+$-128F / SLH-DSA-128F	SPHINCS$^+$-192F / SLH-DSA-192F	SPHINCS$^+$-256F / SLH-DSA-256F
	MAYO	MAYO1	MAYO3	MAYO5
	CROSS	CROSS128	CROSS192	CROSS256

two signature verification operations on the server side and one signature operation on the client side, analogous to the procedure used for server authentication.

3.2. Post-Quantum Algorithms

Traditional cryptographic algorithms are based on mathematical problems which are at risk of being broken by quantum computers. Post-quantum cryptography relies on new classes of problems resistant to quantum attacks[2]. These problems give rise to different categories of post-quantum cryptography algorithms. This section overviews only the types evaluated in this work (see Table 1).

Lattice-Based Cryptography: These schemes rely on the hardness of mathematical problems related to high-dimensional lattices, such as the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP). Several post-quantum algorithms, including three of the four finalists of the post-quantum competition organized by NIST, are based on lattice-based.

ML-KEM [30] is a key-encapsulation mechanism (KEM) standard based on CRYSTALS-KYBER. Its security relies the hardness of the Module Learning With Error (MLWE) problem, which involves solving a system of noisy linear equations in a polynomial ring. In particular, a public key encryption scheme is constructed using a noisy linear equation system where the public key is a randomly generated matrix A and a vector $t = As + e$, and s is the private key. To encrypt a message it is encoded as an additional noisy equation, which can be recovered using the private key. This scheme is transformed into a KEM using the Fujisaki-Okamoto (FO) transform [31] resulting in a KEM believed to provide IND-CCA security [32].

ML-DSA [33] is a standard signature scheme based on CRYSTALS-DILITHIUM. Similar to ML-KEM, it derives its security from the hardness of MLWE but it also relies on the hardness of a variant of the Module Short Integer Solution (MSIS) problem. ML-DSA constructs a Schnorr-like signature from an interactive proof of knowledge protocol, which is made non-interactive by applying the Fiat-Shamir heuristic. In this protocol, the signer demonstrates knowledge of two secret vectors s_1 and s_2 with small coefficients to a verifier who has access to a matrix A and $t = As_1 + s_2$. The difficulty of finding these vectors, linked to the MSIS problem, acts as a trapdoor, enabling the efficient signing of messages.

FALCON [34], another finalist of the NIST competition, is a lattice-based digital signature scheme based on the GPV framework [35]. Similar to ML-DSA, FALCON is also based on the hardness of the MSIS problem but over NTRU lattices (NTRU-SIS). FALCON's private key consists of four small polynomials which can be seen as a good basis of an NTRU lattice, and the public key is bad basis of the same lattice defined by a single polynomial that is mathematically related to the private key. To sign a message, the signer first hashes the message into a polynomial. Then produces two short polynomials s_1 and s_2 satisfying a given equation associated with the hash and the public key. Finding these such short vectors is difficult under the MSIS assumption unless the private key is known. Recent research [36] argues that FALCON lacks a complete security proof, which was thought to follow directly from GPV framework under the NTRU-SIS assumption.

Hash-based Cryptography: This class of algorithms rely on cryptographic hash functions to create post-quantum digital signatures. While there is no established proof of their resistance to quantum computers, it was demonstrated that these functions are both necessary and sufficient for ensuring the security of digital signatures [37]. An additional benefit is that these hash-based signature schemes can be constructed by simply changing to a new cryptographic hash function, avoiding complex problems in number theory or algebra. This concept is not new, with the first schemes dating back to the late 1970s. In fact, the current SLH-DSA standard builds upon ideas from already existing hash-based signature schemes.

SLH-DSA [38] is a post-quantum signature standard based on SPHINCS$^+$. This signature algorithm is based on a XMSS (eXtended Merkle Signature Scheme) [39] hyper-tree structure. This hyper-tree is used to sign the public keys of FORS (Forest of Random Subset) [40] instances, which

[2]No known quantum algorithm provides a significant advantage in solving these problems, but future breakthroughs remain possible.

in turn are used to sign message digests. The rest of layers consists of trees of one-time signatures (WOTS+) [41]. A signature in SLH-DSA consists of a randomization value, a FORS signature of the message digest and a hyper-tree signature of the FORS public key used. Signature verification can be summarized as recomputing message digest, computing a candidate FORS public key, and verifying the hyper-tree signature on that public key. Two variants of this scheme exist: one for fast signature generation ('f') and one for small signature generation ('s').

In 2023, NIST launched a new round of post-quantum signature algorithm [42], driven by two key considerations. First, due to recent advancements in lattice-based cryptosystems and the limited understanding of their resilience against quantum attacks, this round sought proposals for signature schemes that do not rely on structured lattices. Second, given that the standardized hash-based scheme produces relatively long signatures and has a slow verification process, this round aimed to identify alternatives with shorter signatures and faster verification.

Multivariate Public-Key Cryptography: This class relies on the hardness of solving multivariate quadratic (MQ) equations over a finite field. The general problem of solving such a set of equations is NP-hard. One of the earliest multivariate quadratic signature schemes is the Oil and Vinegar (OV) scheme [43]. The idea was hiding quadratic equations in o unknowns called "oil" and $v = o$ unknowns called "vinegar" over a finite field, with linear secret functions. An Unbalanced Oil and Vinegar (UOV) scheme [44] improves security by using more vinegar variables than oil variables. While UOV provides good performance and relatively small signature sizes, its main drawback lies in its large key size.

MAYO [45] is a signature scheme presented to the NIST competition for the standardization of additional post-quantum digital signatures. This scheme is a variant of the UOV scheme that reduces the size of public keys. This is achieved by using very small oil space, which allows for a very compact representation of the public key. However, this reduction comes at the cost of increased signature size and a more computationally intensive signature process. The security of MAYO is conditioned to the hardness of UOV problem and the a new tailored problem called the Multi-Target Whipped MQ (MTWMQ) problem. The hardness of the latter needs to be studied further ensure the security of the scheme.

Code-based Cryptography: These schemes use error-correcting codes as the foundation for cryptographic primitives. One of the earliest code-based cryptographic scheme was a public key encryption schemes proposed by Robert J. McEliece [46]. This scheme uses a binary irreducible Goppa code –an error-correcting code defined by a polynomial over a finite field– as a private key, and a randomly permuted generator matrix of that code as the public key. A ciphertext consists of a codeword with added errors, which can only be corrected by the holder of the Goppa code.

CROSS [47] is a code-based signature scheme submitted to the second call for post-quantum signature proposals

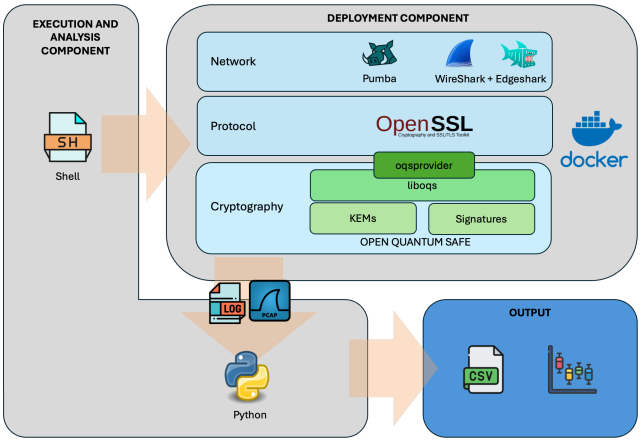


Figure 2: Performance Evaluation Framework

launched by NIST. The scheme is based on a proof of knowledge for the Restricted Syndrome Decoding Problem (R-SDP), which is closely related to the classical SDP problem [48]. In CROSS, the signer chooses a random parity-check matrix generated from a seed and a syndrome –a vector used to detect errors– which is based on the matrix and a private vector. The protocol involves two commitments, one to prove the syndrome equation and another one to prove the restriction on the private vector. Additionally, two challenges are involved: the first is the hash of the message and the two commitments, and the second also incorporates the first challenge and responses. The protocol is made non-interactive using the Fiat-Shamir heuristic. A CROSS(G) variant uses a particular subgroup for the private vector for improved efficiency and smaller key sizes.

4. TLS Performance Evaluation Framework

This section presents our evaluation framework[3], designed to facilitate the evaluation and analysis of TLS configurations employing classical, post-quantum, and hybrid cryptographic primitives under both ideal and realistic network conditions. The framework follows a modular architecture, consisting of two main components, as depicted in Figure 2.

Deployment component: This component includes all the tools and libraries necessary for the deployment of scenarios, organized into three functional layers. These layers are encapsulated within a containerized environment to enable seamless deployment of client and server machines across different platform architectures. Moreover, the use of containers ensures a consistent and reproducible working environment for all evaluations. For this purpose, we rely on `Docker` as the underlying container technology.

The first functional layer is responsible for controlling the network between the client and the server. On the one hand, it enables the definition of diverse scenarios, allowing

[3] Available at the GitHub repository: `https://github.com/montenegro-montes/TLS-PQ/`

the specification of network parameters at the container level, including packet loss and delay. This is achieved using a flexible and comprehensive tool called `Pumba`. On the other hand, this layer allows network traffic analysis by integrating `EdgeShark`, a tool that enables `Wireshark` to capture packets at the container level.

The second layer comprises the tools and configurations required to evaluate the TLS protocol. In this work, it integrates the latest available version of `OpenSSL` (3.3.1) at the time of testing, although other versions or libraries may be integrated instead.

The third layer provides the cryptographic primitives required to perform post-quantum and hybrid TLS handshakes. By default, the version of `OpenSSL` used only provides traditional primitives. This layer incorporates the necessary libraries and configuration changes to enable the use of additional cryptographic providers. Specifically, it integrates the provider from the Open Quantum Safe (OQS) project, `oqsprovider`, which enables access to the post-quantum and hybrid primitives implemented in the `liboqs` library (version 0.12.0).

Execution and Analysis component: This component manages the execution of specific scenarios and TLS configurations through a set of `Shell` scripts organized by security levels. There are two types of scripts. The first type assesses TLS handshake performance by measuring the number of handshakes completed in 10 seconds; each configuration and scenario is executed 50 times to ensure statistical significance. The second type evaluates communication overhead by measuring the kilobytes transmitted per TLS handshake configuration and scenario. A single execution is sufficient in this case, as traffic size remains consistent for each configuration.

Each execution produces client and server logs, along with packet capture (`PCAP`) files generated by traffic analysis tools. These files are automatically processed with `Python` scripts to extract the most relevant data into comma-separated values (`CSV`) files, which are then used to generate plots for each evaluated scenario. Results are shown using bar plots, while box-and-whisker plots (boxplots)— displaying quartiles, median, and outliers—are included for tests executed multiple times.

5. Evaluation of cryptographic primitives

Prior to evaluating the performance of TLS, this section examines the performance of the individual KEM and signature primitives involved in the TLS handshake. The evaluation is based on OpenSSL's `speed` tool.

The following command enables benchmarking the KEM algorithms available in TLS. The command executes each KEM operation (key generation, encapsulation and decapsulation) for a specific number of seconds:

```
openssl speed -kem-algorithms -elapsed -mlock
              -seconds <sec>
```

Table 2
KEM operations performed in 10 seconds

Security Level	KEM Primitive	Keygen (ops)	Encaps (ops)	Decaps (ops)
I	P-256	483794	62035	71719
	X25519	212445	101879	203787
	p256_mlkem512	10141	48745	18897
	x25519_mlkem512	133930	84475	86224
	mlkem512	537249	812861	916541
III	P-384	9369	3522	5672
	X448	35780	19838	44380
	p384_mlkem768	4813	3393	4505
	x448_mlkem768	27703	16859	17005
	mlkem768	368546	540278	573492
V	P-521	3970	1553	2541
	p521_mlkem1024	2724	1488	2195
	mlkem1024	288839	358368	370072

Table 3
Traditional signature operations in 10 seconds and certificate size in bytes

Security Level	Signature Primitive	Operations		Certificate Size (bytes)
		Sign	Verify	
I	Ed25519	224405	74452	468
III	secp384r1 (ecdsap384)	8651	10433	634
V	secp521r1 (ecdsap521)	3680	4729	740

The results of executing this command are presented in Table 2 organized by security levels. The best performing KEM primitive for all operations and security levels is ML-KEM. It demonstrates significantly better performance compared to the hybrid (i.e., combining traditional and post-quantum) primitives recommended by international institutions [49]. However, this recommendation is primarily motivated by concerns regarding the reliability of their novel security primitives rather than their performance.

The speed tool can also be used to benchmark the signature algorithms available in TLS by executing the following command:

```
openssl speed -signature-algorithms -elapsed -mlock
              -seconds <sec>
```

The result of this command are divided in three tables. Table 3 presents the number of signature and verification operations for traditional signature primitives in Levels I, III and V. This table serves as the baseline for comparing with hybrid and post-quantum signature primitives, shown in Tables 4 and 5, respectively. The tables also include the certificate size in bytes.

The results in Table 4 indicate that the only hybrid signature primitive that exhibits a performance that is comparable to traditional primitive at Level I is `p256_mayo`. However, it is only comparable in terms of verifications, with `p256_mayo` being even slightly superior to `ed25519`. In contrast, the number of signature operations is reduced by approximately 3.5 times. Despite the fact that the size of the certificates increases six-fold in comparison with the traditional option, it is the most effective of the level I options. The second-best

Table 4
Hybrid signature operations in 10 seconds and certificate size in bytes

Security Level	Hybrid Signature Primitive	Sign	Verify	Certificate Size (bytes)
I	p256_falcon512	21906	69523	2649
	p256_sphincssha2128fsimple	828	8619	23730
	p256_sphincssha2128ssimple	42	18536	11229
	p256_mayo1	67365	88501	2570
III	p384_mldsa65	7298	8725	7760
	p384_sphincssha2192fsimple	543	4318	48998
	p384_sphincssha2192ssimple	26	6406	22673
	p384_mayo3	6889	9271	5018
V	p521_mldsa87	3403	4402	10510
	p521_falcon1024	2758	4406	4876
	p521_sphincssha2256fsimple	260	2870	68336
	p521_sphincssha2256ssimple	28	3318	41166
	p521_mayo5	3009	4371	8654

Table 5
Post-quantum signature operations in 10 seconds and certificate size in bytes

Security Level	PQ Signature Primitive	Sign	Verify	Certificate Size (bytes)
I	falcon512	23678	167563	2450
	sphincssha2128fsimple	843	9064	23535
	sphincssha2128ssimple	42	20253	11034
	mayo1	88684	346312	2373
	CROSSrsdpg128balanced	10540	18163	12952
III	mldsa65	60863	154796	7504
	sphincssha2192fsimple	576	7575	48710
	sphincssha2192ssimple	26	15509	22385
	mayo3	30732	90506	4738
	CROSSrsdpg192balanced	6833	11660	32147
V	mldsa87	48759	99038	10153
	falcon1024	11961	82157	4506
	sphincssha2256fsimple	278	7104	67954
	sphincssha2256ssimple	28	11638	40784
	mayo5	15283	44191	8276
	CROSSrsdpg256balanced	3947	6835	54865

hybrid option is `p256_falcon256`, but its primary drawback is a tenfold reduction in the number of signatures operations.

At Level III, `p384_mldsa65` and `p384_mayo3` demonstrate comparable performance to the traditional `ecdsap384` primitive, exhibiting a modest reduction in the number of signature and verification operations. The main downside of these hybrid primitives is the approximately tenfold increase in certificate size. Notably, the performance of the only traditional signature primitive available at Level V, `ecdsap521`, is comparable to most hybrid primitives, except for those based on SPHINCS$^+$, which remain significantly less efficient across all security levels, even in its fast variant. At this level, certificate size remains a concern, with `p521_falcon1024` introducing the least overhead, with almost 5 kilobytes in size, which is approximately 6.5 times bigger compared to a traditional `ecdsap521` certificate.

The results of the evaluation of post-quantum signatures primitives are shown in Table 5. This table incorporates the primitive MAYO, which at the time of writing lacks of a hybrid implementation in liboqs. Among the various versions of this primitive available, the balanced version has been selected for testing. At Level I, all the primitives demonstrate inferior performance compared to traditional counterparts, with MAYO representing the most favorable case, reducing the number of signatures by a factor of 2.5. In contrast, for verification, both FALCON and MAYO outperform the traditional primitive, achieving 2.3 and 4.5 times more verifications, respectively. Additionally, these options introduce the smallest increase in certificate size.

At Level III, both ML-DSA and MAYO demonstrate higher performance compared to the traditional option, particularly in terms of signature with verification. However, they impose a significant cost on the certificate size, increasing it by 12 and 7.5 times, respectively. A similar trend is shown at Level V, with the addition of FALCON, which also provides a boost in performance with respect to traditional primitives. While FALCON is not as efficient as MAYO and ML-DSA, it is the one with the smallest certificate size - only 6 times larger than a traditional certificate. The least performing primitive at all levels is SPHINCS$^+$, which is capable of performing only very few signature operations per second – from 2 to 4 operations in the small variant and from 30 to 80 in its fast variant. In terms of certificate size, is also the worst performing option with CROSS being right in between the small and fast variants.

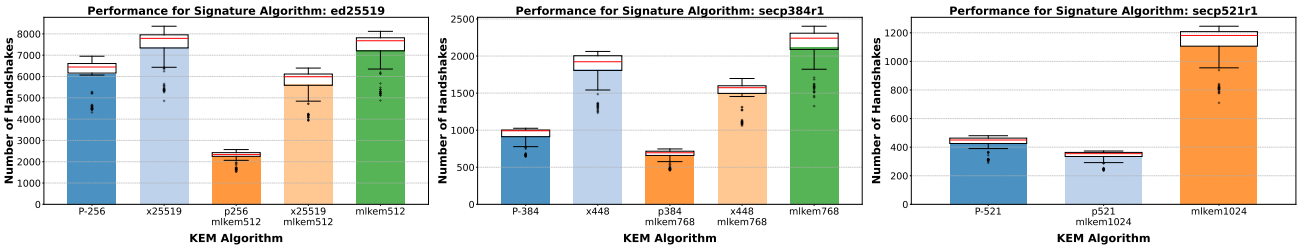
6. Current Status of Post-Quantum TLS

Current efforts to incorporate post-quantum primitives in TLS focus on the key exchange phase, while the authentication phase still remains on a pre-quantum state. This decision is primarily motivated by the cost associated with post-quantum signatures (as shown in Sec. 5). Moreover, the most pressing threat is the 'harvest now, decrypt later' problem, which affects the key exchange phase. However, given the novelty of post-quantum KEMs, major international agencies (e.g. [50]) are recommending the use of hybrid KEM algorithms.

This section examines the current status of the TLS protocol in the context of the recently established post-quantum cryptography standards. Specifically, it analyzes the computational and transmission overhead associated with these new cryptographic primitives. The primary objective is to identify potential challenges and limitations in the transition towards a quantum-safe web.

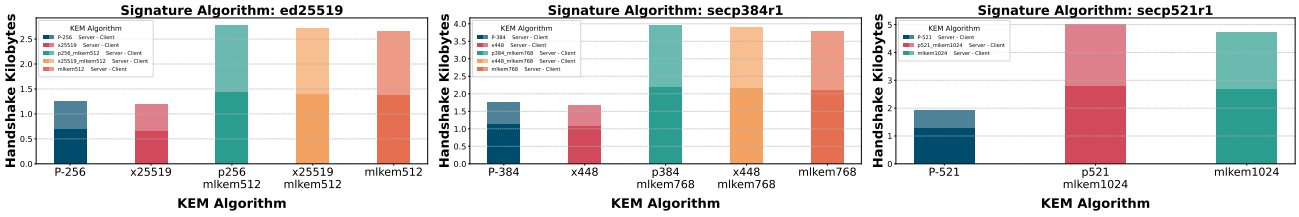
Post-quantum primitives can be introduced in the handshake phase of TLS to secure the key exchange and authentication processes. For the key exchange, there is only one possible post-quantum standards, that is ML-KEM, but there are currently two alternatives for authentication, namely, ML-DSAand SLH-DSA.

The performance evaluation of various Key Exchange Mechanisms (KEM), including traditional, post-quantum, and hybrid approaches, is illustrated in Figure 3 and 4. These figures are organized based on the security levels of the cryptographic primitives employed during the handshake phase. To focus specifically on the performance of the KEM algorithms, the signature algorithm used in the authentication process is fixed. For each security level, an efficient traditional signature algorithm is selected, ensuring that the



(a) Level I (b) Level III (c) Level V

Figure 3: Number of TLS handshakes in 10 seconds.



(a) Level I (b) Level III (c) Level V

Figure 4: Number of kilobytes transmitted during a TLS handshake.

evaluation centers solely on the performance characteristics of the KEM.

Figure 3 shows the number of complete TLS handshakes performed in a time period of 10 seconds. In Figure 3a, the performance of fully post-quantum and hybrid post-quantum algorithms are compared to two traditional KEM approaches based on elliptic curves, NIST P-256 and Curve 25519. It can be observed that a TLS configuration based on traditional algorithms is not significantly better than one using ML-KEM. In fact, the performance of standalone ML-KEM is better than P-256 and similar to Curve 25519, which is the most efficient KEM at this security level. Hybrid schemes are the least efficient ones since they combine elliptic curves with ML-KEM. However, using traditional P-256 for KEM is only slightly better than a hybrid approach of Curve 25519 and ML-KEM. The results in Figure 3b show a similar trend with a reduction of around one third in number of connections. Interestingly, in this level, the most efficient approach is fully post-quantum and the hybrid schemes is better than the traditional KEM based on NIST P curve. Finally, Figure 3c shows that the post-quantum scheme outperforms the traditional KEM based on P-521, and the performance of the hybrid scheme is similar to the traditional one.

Figure 4 depicts the transmission overhead imposed by the same TLS configurations analyzed in the previous figure for a single handshake. In this figure, we distinguish between the amount of kilobytes transmitted by the client and the server in order to identify any significant communication imbalance. As illustrated in Figure 4a, traditional algorithms exhibit greater bandwidth efficiency compared to hybrid and post-quantum KEMs. While the best and worst configurations remain consistent with the analysis of the number of connections, the differences between algorithms are less pronounced. Both traditional algorithms perform similarly, transmitting less than 1.5 KBytes per handshake. In contrast, all configurations involving post-quantum KEMs increase the amount of data transmitted per handshake by a factor of approximately 2.2 to 2.4. While similar differences are observed between traditional and post-quantum KEMs at security level III (see Figure 4b), in the worst case, the amount of kilobytes transmitted is approximately 4 kilobytes. Finally, Figure 4c shows that TLS configurations with post-quantum KEMs at security level V result in about 5 kilobytes of data per handshake, compared to slightly less than 2 kilobytes for traditional configurations.

In conclusion, the post-quantum ML-KEM emerges as the best-performing KEM approach for TLS, particularly at higher security levels where its efficiency becomes more pronounced. In contrast, hybrid approaches are the least efficient, especially when ML-KEM is combined with NIST curves. However, hybrid KEMs using Curves 25519 and 448 exhibit performance that is comparable to, or even better than, traditional approaches based on NIST curves. In terms of bandwidth, traditional KEM approaches consistently outperform post-quantum methods. Notably, the transmission overhead between ML-KEM and hybrid approaches is minimal, indicating that the choice of hybrid configurations does not significantly increase data transmission requirements.

Evaluating the algorithms across different levels and conducting a cross-level study can be highly beneficial. This approach helps identify the cost of transitioning from one security level to another, which typically results in a decrease in connections and an increase in information transmission. Understanding these costs is crucial for decision-makers who need to balance security requirements with system performance.

Table 6
TLS handshakes using traditional (T) or hybrid (H) KEM primitives. All configurations use secp384r1 for authentication.

Handshake	T/H	KEM_ALG_CLIENT	KEM_ALG_SERVER	Transmitted Bytes	
				Server	Client
Regular	T	x25519	x25519	832	487
		X25519MLKEM768:x25519_kyber768:x25519	x25519		
	H	x25519_kyber768	x25519_kyber768	1921	1663
		X25519MLKEM768	X25519MLKEM768		
HRR	T	x25519_kyber768:x25519	X25519MLKEM768:x25519	936	2082
		x25519	X25519MLKEM768:x25519_kyber768:x25519		
	H	x25519_kyber768:x25519	X25519MLKEM768:x25519_kyber768:x25519	2013	3270
		x25519_kyber768	X25519MLKEM768:x25519_kyber768		
Error	–	x25519_kyber768	X25519MLKEM768	7	1583
		X25519MLKEM768	x25519_kyber768		
		x25519	x25519_kyber768		
		x25519	X25519MLKEM768		

By selecting the optimal option at each level, the number of connections decreases from 7356 at Level I to 2115 at Level III, which represents a 71.25% reduction. At Level V, the number of connections further drops to 1109, marking an 84.92% decrease from Level I. The shift from Level III to Level V results in a 47.57% reduction in connections.

In contrast, data transmission increases significantly across levels. From Level I to Level III, the transmitted bytes rise from 701 to 2138, a 204.99% increase. Moving from Level III to Level V, the bytes increase from 2138 to 2730, representing a 27.69% rise. Overall, from Level I to Level V, the data transmission grows from 701 to 2730 bytes, resulting in a 289.44% increase.

7. Additional challenges

The results presented in the previous section were obtained under ideal network conditions to provide insights into the maximum performance of various TLS configurations based on both traditional and post-quantum primitives, ensuring a fair comparison between them. However, in real-world scenarios, several challenges are likely to arise when deploying TLS configurations with PQC primitives. In this section, we examine four key challenges using our evaluation framework.

7.1. Hello Retry Request

A typical TLS handshake starts with the client sending a `ClientHello` message where it indicates to the server, among other parameters, a list of KEM mechanisms supported by the client in descending order of preference [29]. Additionally, the client may add a key share (i.e., cryptographic parameters) for some or all of these groups.

In case the client and server are unable to agree on a common set of parameters, the TLS handshake terminates, and the connection is closed. However, in certain scenarios, the server may identify an acceptable set of parameters but requires additional information from the client to proceed. This situation often arises when the client proposes a specific KEM in the `ClientHello` message but does not include the corresponding key share.

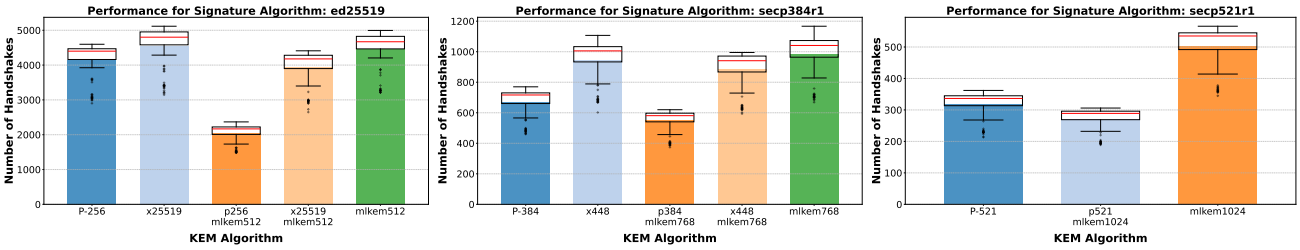
In such situation, the server sends a `HelloRetryRequest` (HRR) message [1], requesting the client to provide a key share for the mutually supported KEM. Upon validating the HRR, the client responds with an updated `ClientHello` message containing the requested key share. While necessary, this process increases the communication overhead.

To evaluate the transmission overhead imposed by this process, we defined a series of client and server TLS configurations using traditional and hybrid KEMs. These include regular configurations, where the handshake proceeds normally; configurations where the client and the server have a KEM in common but HRR is required; and configurations where they support different KEMs and result in error. As shown in Table 6, a regular handshake using Curve 25519 results in a total transmission of 1319 bytes (of which 487 bytes are transmitted by the client and 832 by the server), while a negotiation involving a hybrid KEM implies 3584 bytes. When the client does not send a key share for the mutually supported KEM, the need for HRR results approximately in 2.3 and 1.5 increase, respectively, even when the actual primitives used are the same. Finally, even if the client and server configurations mismatch and result in error, it requires a transmission over 1.5 kilobytes.

In addition to the increased transmission overhead, a more concerning scenario deserves attention. A state-of-the-art client configured to use the hybrid `X25519MLKEM768` as its preferred KEM may inadvertently fall back to the traditional `x25519`, a quantum-insecure KEM, which is configured as a last-resort option. This situation can occur even if the server is configured to support hybrid post-quantum KEMs but instead employs the pre-standard `x25519_kyber768` KEM.

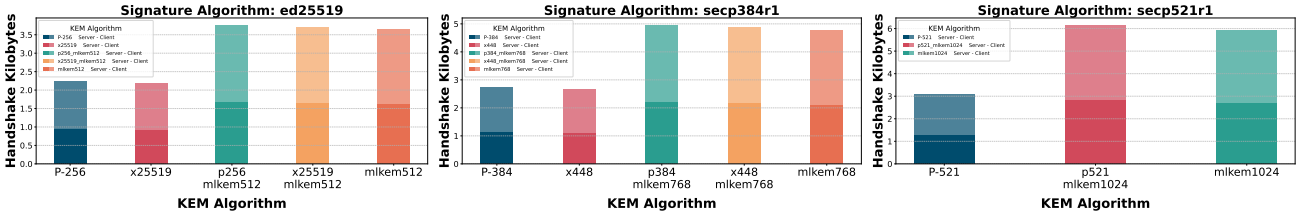
To mitigate this issue, the client could simply avoid configuring the use of the traditional `X25519` KEM. However, this approach might lead to frequent handshake failures when communicating with servers that do not handle sensitive data. For a smoother and safer transition, web browsers should notify users when they are configured to use hybrid or post-quantum KEMs, but the negotiation results in a quantum-insecure handshake. As these are low-level protocol settings that most users are not familiar with and may pose a security risk, it is recommended to alert users through a warning message. The warning could take the following form:

"Warning: The TLS configuration of the server you are connecting to is not resistant to quantum attacks. This



(a) Level I (b) Level III (c) Level V

Figure 5: Number of mutually authenticated TLS handshakes in 10 seconds



(a) Level I (b) Level III (c) Level V

Figure 6: Number of kilobytes transmitted during a mutually authenticated TLS handshake.

communication could be intercepted and decrypted in the future using quantum technologies."

This approach ensures that users are informed of potential security implications when post-quantum primitives are not employed.

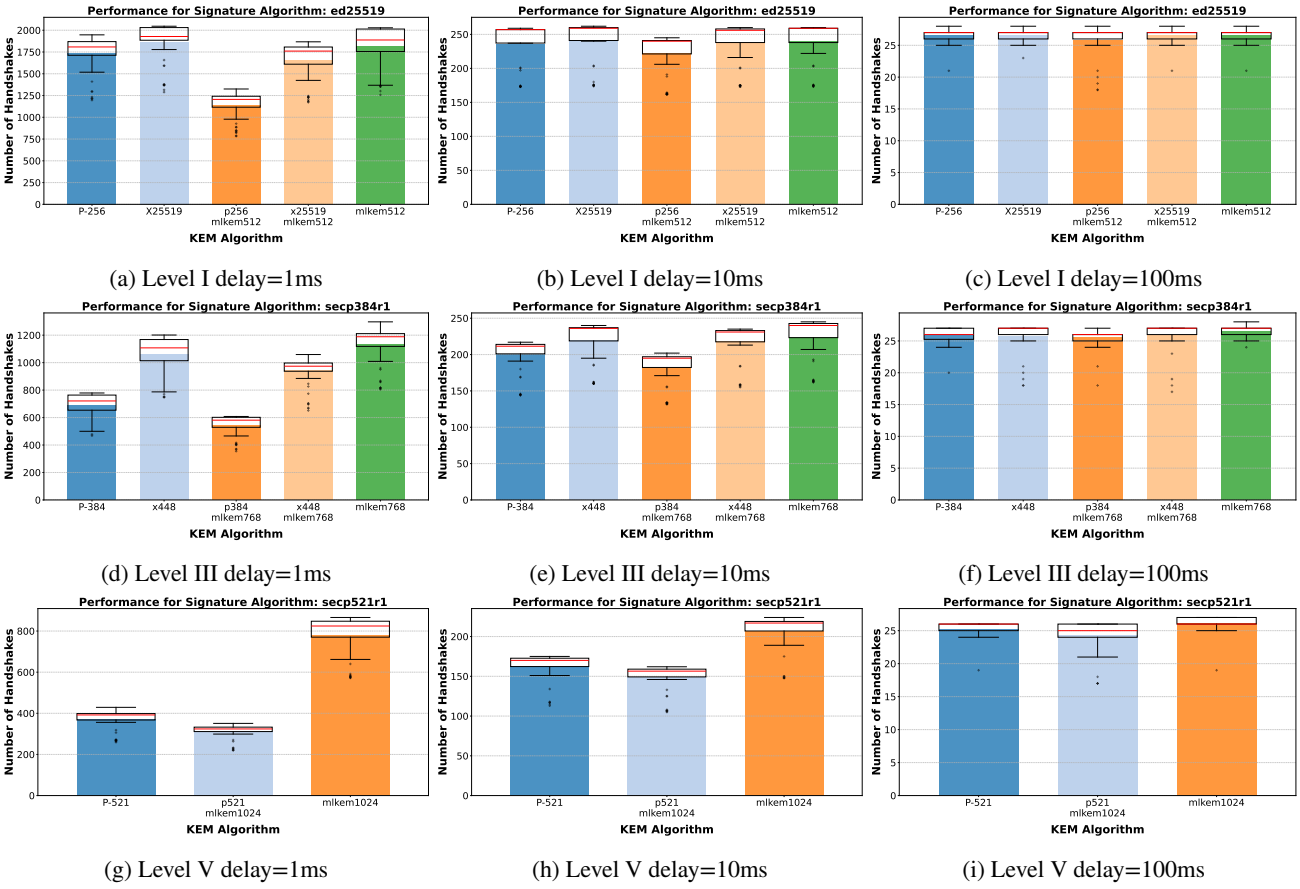
7.2. Mutual Authentication

While most web connections rely on server authentication to allow the client verify that it is connected to the intended server, there are critical scenarios - such as handling administrative procedures with public authorities, where the server must also verify the client's identity. It also allows for connections without a login process, which is useful for non-human users such as Internet of Things (IoT) devices.

The mutual authentication process is triggered in TLS when the server sends a `CertificateRequest` message to the client, which contains a set of extensions describing the features that the client's certificate must satisfy to be accepted, including the signature algorithms of the certificate. Upon reception of this message, the client transmits a `Certificate` message that contains the requested certificate, and a `CertificateVerify` message. The latter message contains a signature of the entire handshake using the private key associated with the client's certificate. Consequently, mutual authentication in TLS introduces additional computational and transmission overhead. The extent of this overhead depends on the length of the certification chain and the specific signature algorithm used. In particular, validating the client's certificate is a recursive process that involves verifying the signature of each certificate in the certification chain, all the way up to the root certificate authority (CA)[4]. Additionally, the server must verify the validity of signature in the `CertificateVerify` message.

The results of our evaluation of mutually authenticated TLS are presented in Figures 5 and 6. Regarding the number of connections (Figure 5), it is worth noting that the performance decrease varies across configurations. For instance, hybrid configurations involving a NIST P curve and ML-KEM show a performance reduction of approximately 8%, 17%, and 19% in Levels I, III, and V, respectively. In contrast, the number of connections is reduced by half in Level III and V ML-KEM configurations. Additionally, the performance decrease compared to server-only authentication ranges from 30% to 38% at Level I, 27% to 48% at Level III, and 19% and 25% at Level V. The reason behind a more pronounced performance detriment at lower security levels is due to the higher cost of these primitives (c.f. Section 5). At high security levels operations are more costly, therefore the introduction of an additional round of traditional signature and verifications for client authentication does not impact as much compared to lower security levels. The same applies when comparing configurations at the same security level. When the primitive is computationally intensive, the overhead introduced by the additional authentication round has a more pronounced effect in percentage terms. In terms of traffic volume (Figure 6), the transition from server-side to mutual authentication can be summarized as approximately adding 1 kilobyte of data for all primitives and levels. This conclusion is drawn by comparing Figures 4 and 6, and this

[4] In our tests, the client's certificate is signed directly by the root CA. As a result, two signature verifications are required for validating the client's certificate.



(a) Level I delay=1ms (b) Level I delay=10ms (c) Level I delay=100ms

(d) Level III delay=1ms (e) Level III delay=10ms (f) Level III delay=100ms

(g) Level V delay=1ms (h) Level V delay=10ms (i) Level V delay=100ms

Figure 7: TLS Handshake connections during 10 seconds Levels I,III and V for various delays.

increase is mostly due to the transmission of the client's certificate.

Interesting insights are also revealed by comparing the results across security levels. First, we examine the performance degradation associated with increasing the security level. Choosing the best performing configuration at each level, the number of TLS connections drops from 4554 at Level I to 983 at Level III, which represents a 78.41% reduction. At Level V, the number of connections further drops to 501, marking an 89.00% decrease from Level I. The shift from Level III to Level V results in a 49.03% reduction in connections. In terms of the data transmission volume, the increase is significant across levels. From Level I to Level III, the number of received bytes rise from 1345 to 2746, a 104.16% increase. At Level V, the number of bytes transmitted is 3316, representing a 20.76% rise. Overall, from Level I to Level V, the transmission overhead grows from 1345 to 3316 bytes, resulting in a 146.54% increase.

7.3. Realistic Network Conditions

The results presented and analyzed in previous sections consider ideal network conditions. However, packet losses and delays are inevitable in real-world scenarios. In this section, we explore the impact of these conditions on the performance of various TLS configurations.

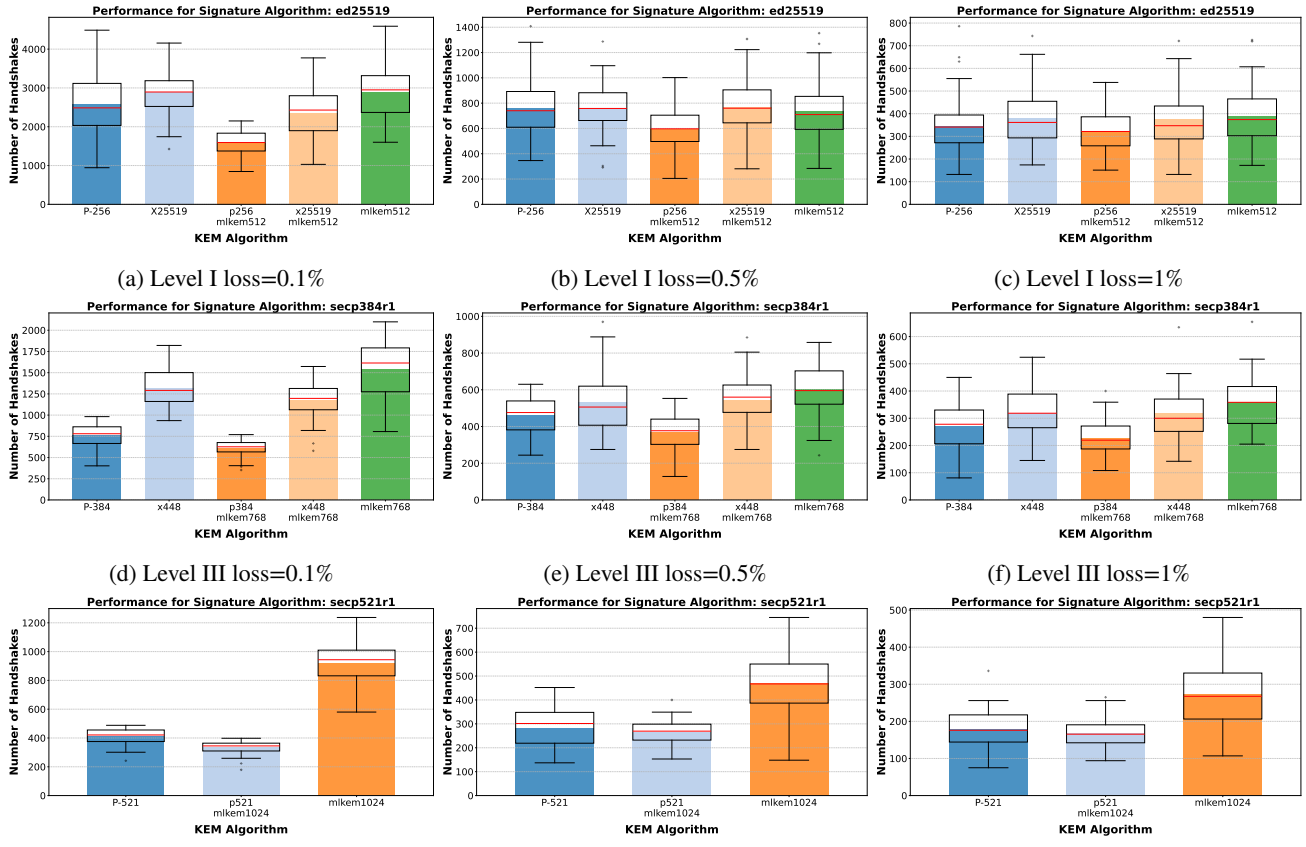
7.3.1. Packet Delay

This section explores the impact of different packet delays on several KEM configurations at different security levels. Delays of 1, 10 and 100 milliseconds[5] are considered for testing. The results are illustrated in Figure 7.

At Level I (Figures 7a, 7b and 7c), the number of connections decreases by 70% with a delay of 1 ms, 90% with a 10 ms delay, and up to a 99% with 100 ms delay. At Level III, the performance degradation is less pronounced (Figures 7d, 7e and 7f) ranging from 17% to 46% for a 1 ms delay, 72% to 89% for a 10 ms delay, and up to 97% for a 100 ms delay. At Level V, the impact of delays is comparatively minor (Figures 7g, 7h, and 7i), with degradation ranging from 7% to 30% for a 1 ms delay, and up to approximately 95% fo a 100 ms delay.

In general, the results indicate that packet delays significantly impact TLS performance, irrespective of the security level. Specifically, packet delays tend to normalize the number of connections that can be established in a given period of time. Notably, when packet delays are sufficiently high, say 100 ms, all TLS configurations exhibit a similar performance, with approximately 25 connections over a 10-second time period. This behavior can be attributed to the fact that

[5] As of 2024, the average latency for fixed broadband is 9 ms and 26 ms for mobiles. See https://www.speedtest.net/global-index



(a) Level I loss=0.1% (b) Level I loss=0.5% (c) Level I loss=1%

(d) Level III loss=0.1% (e) Level III loss=0.5% (f) Level III loss=1%

(g) Level V loss=0.1% (h) Level V loss=0.5% (i) Level V loss=1%

Figure 8: TLS Handshake connections during 10 seconds Levels I, III and V for various loss rates.

the execution time of KEM primitives is substantially lower than the delay introduced.

7.3.2. Packet Loss

This section analyzes the impact of uniform packet loss on TLS performance. We evaluate three loss values of 0.1%, 0.5%, and 1% based on typical Internet conditions[6]. The results are presented in Figure 8.

At Level I (Figures 8a, 8b and 8c), most KEM primitives exhibit approximately a 60% reduction in the number of connections when a 0.1% packet loss rate is introduced. The value increases to 85% at a 0.3% loss rate and reaches 94% at a 1% loss rate. On average, the number of connections ranges from 322 to 390 with hybrid `p256_mlkem512` being the least performing solution. At Level III, the number of connections is reduced in a lower rate. The reduction ranges from 7% to 27% for loss rate of 0.1% (Figure 8d). The reduction ranges from 44% to 72% at a 0.5% loss rate (Figure 8e) and up to 82% reduction at a 1% loss rate 8f). At this level, NIST `P-384` and `p384_mlkem768` are the least performing algorithms but the difference is reduced as the loss rate increases. At Level V (Figures 8g, 8h and 8i), the post-quantum `mlkem1024`, is the KEM that experiences the greatest performance degradation, reaching up to 75% at a 1% loss rate. However, it still doubles the number of connections compared to other primitives.

In summary, packet loss degrades the performance of TLS but not as much as packet delay. As the packet loss rate grows, it also tends to homogenize the number of connections between primitives. However, not all levels are affected equally. On average, the number of connections with Level I primitives almost doubles the number of connections at Level V.

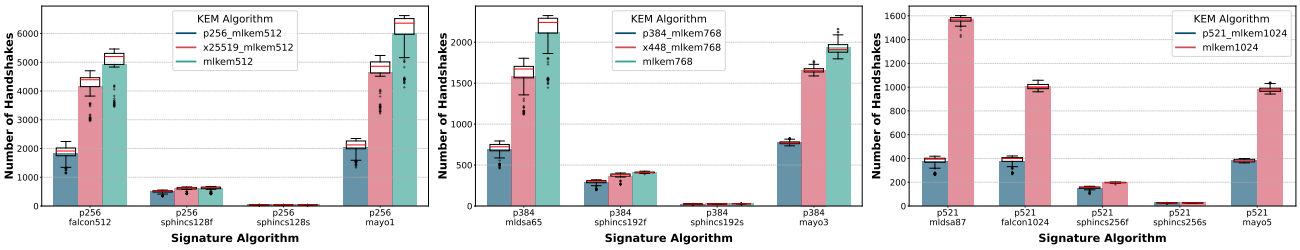
8. Future Post-Quantum TLS

The natural progression toward a quantum-resistant web involves incorporating post-quantum signature algorithms into the TLS authentication phase. For a gradual transition, as with the KEM, we anticipate the integration of hybrid signatures and, ultimately, post-quantum signatures.

Among the existing post-quantum signature algorithms, our evaluation focuses on algorithms standardized after the NIST competition, namely, ML-DSA and SLH-DSA, and, FALCON, which was selected but not standardized. The reason for including the latter algorithm is its recommendation by several international agencies.

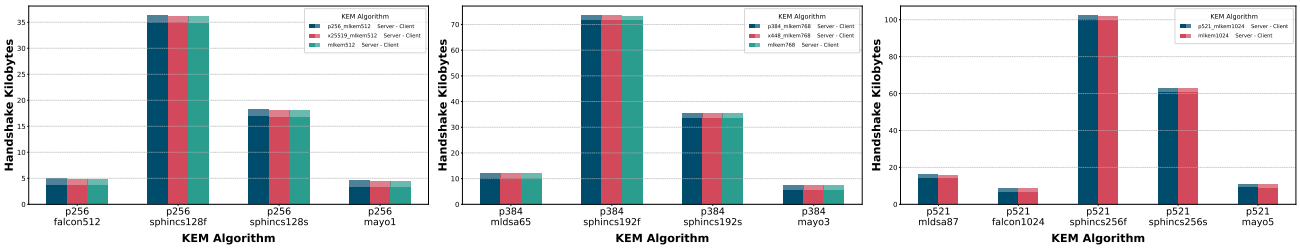
After the first selection of post-quantum algorithms, NIST launched a new competition round to incorporate additional signature algorithms based on different mathematical

[6] As noted in RFC 7680 [51], "healthy Internet paths should be operating at loss ratios below 1%".



(a) Level I (b) Level III (c) Level V

Figure 9: Number of TLS handshakes with hybrid signatures in 10 seconds



(a) Level I (b) Level III (c) Level V

Figure 10: Number of kilobytes transmitted during a TLS Handshake with hybrid authentication

problems. From the numerous new proposals, only MAYO and CROSS are currently implemented in the Open Quantum Safe library. These algorithms have also been included for evaluation.

8.1. Hybrid signatures

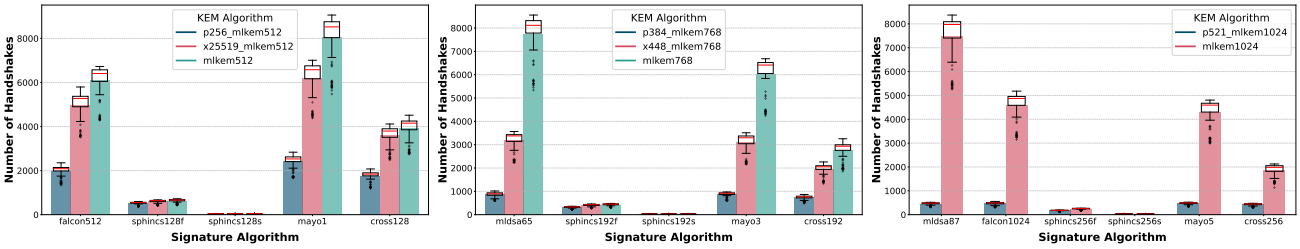
A hybrid signature combines both classical and post-quantum signature algorithms to ensure resistance against both classical and quantum computing attacks. To maintain consistency across all security levels, the hybrid signatures selected for evaluation are based on a NIST P curve and one of the post-quantum algorithms mentioned above.

The results of evaluating hybrid signature algorithms[7] in TLS are shown in Figure 9 and 10. To facilitate comparison with traditional signatures (Sec. 6), TLS configurations using hybrid and post-quantum KEMs are used. Moreover, this approach is consistent with the natural evolution toward a fully post-quantum TLS.

At Level I (Figure 9a), the best performance is achieved by the hybrid versions of MAYO and FALCON. Notably, signatures algorithm performs better when paired with ML-KEM rather than its hybrid versions. Overall, the best performing configuration at this level uses ML-KEM and P256_MAYO1, offering a performance comparable to a traditional configuration using P256 as KEM and Curve 25519 for authentication. At Level III (Figure 9b), P384_MLDSA65 and P384_MAYO3 achieve the highest number of connections, with the latter being approximately 9.4% less efficient. Notably, the choice of a post-quantum KEM offers a 33% performance increase compared to the hybrid versions. Finally, at Level V (Figure 9c), the hybrid version of ML-DSA, P521_MLDSA87, exhibits the best performance, particularly when combined with ML-KEM for key exchange. The hybrid version of FALCON follows with 56% fewer connections, while MAYO shows almost a 60% reduction in performance. At all levels, hybrid versions of SPHINCS$^+$ demonstrate poor performance, as anticipated by Table 4. Notably, moving from Level I to Level III leads to a nearly threefold reduction in connections, while at Level V, the reduction is approximately fourfold. The transition from Level III to Level V is less pronounced, with the number of connections not even halving.

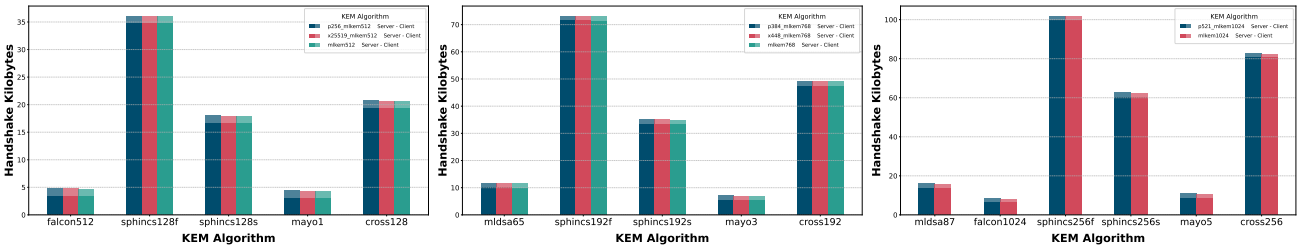
In terms of data transmission volume, at Level I (Figure 10a), the hybrid versions of MAYO and FALCON have similar overhead, both exceeding 3 kilobytes, with P256_FALCON512 transmitting approximately 12% more data. At Level III (Figure 12b), the best option is the configuration with the hybrid version of MAYO, which transmits approximately 5800 bytes, while hybrid ML-DSA transmits just over 10 kilobytes of data. At Level V (Figure 10c), the configuration that the lowest transmission overhead is P521_FALCON1024, with almost 7 kilobytes, followed by P521_MAYO5, with over 9 kilobytes, and P521_MLDSA87, which imposes twice the transmission overhead of hybrid FALCON. Configurations based on hybrid SPHINCS$^+$ are highly inefficient at all levels, transmitting 5 to 12 times more data compared to the best alternatives, with up to 100 kilobytes per connection at Level V. Notably, when selecting the best option at each level, data volume increases by 73% from Level I to Level III, and by 19% from Level III to Level

[7] At the time of writing hybrid versions of SLH-DSAwere not available. Hybrid versions of SPHINCS$^+$are used instead. No hybrid implementations of CROSS are available either.



(a) Level I (b) Level III (c) Level V

Figure 11: Number of TLS handshakes with post-quantum signatures in 10 seconds



(a) Level I (b) Level III (c) Level V

Figure 12: Number of kilobytes transmitted during a TLS Handshake with post-quantum authentication

V, resulting in a total increase of 107% from Level I to Level V.

In summary, hybrid MAYO is the best choice at Level I both in terms of connections and data volume. At Level III, the optimal choice depends on the scenario since hybrid ML-DSA is more connection efficient, but hybrid MAYO minimizes data transfer. The situation is similar at Level V, with hybrid ML-DSA offering the best performance but hybrid FALCON reducing the transmission volume. In all levels, the use of SPHINCS$^+$ is discouraged due to its poor performance and data transfer efficiency.

8.2. Post-quantum signatures

The evaluation of post-quantum signatures follows a trend that remains consistent to that of hybrid signatures across all three security level (see Figures 11 and 12). Specifically, MAYO achieves the highest number of connections at Level I, while ML-DSA is superior at Levels III and V. Regarding the number of transmitted bytes, MAYO is the recommended option at Levels I and III, while FALCON is preferable at Level V. While the results are similar it is worth noting that using post-quantum signatures are far more efficient than using hybrid signatures both in terms of connections and transmission overhead.

In this evaluation, the CROSS primitive is also considered, which was not available in its hybrid form. It can be observed that, in terms of the number of connections, this primitive only performs better than SPHINCS$^+$. However, regarding data transmission volume, CROSS introduces a significant overhead, positioning it between the small and fast versions of SPHINCS$^+$. This implies handshakes of approximately 20 kilobytes of data at Level I and over 80 kilobytes at Level V.

An interesting observation is that the transition between security levels is less costly compared to hybrid signatures. The reduction in the number of connections across levels is not very pronounced. Specifically, transitioning from Level I to Level III results in a decrease from 8052 to 7746 connections, representing a reduction of 3.79%. Similarly, moving from Level III to Level V, the number of connections decreases from 7746 to 7484, corresponding to a 3.39% reduction. Overall, when comparing Level I to Level V, the total reduction is approximately 7%. These results suggest that the performance bottleneck in Figure 9 comes from traditional signature primitives at high security levels. This is corroborated by the evaluation of standalone cryptographic primitives performed in Sec. 5.

The data transmission increases progressively across levels. Specifically, considering the most data-efficient configuration at each level, we observe that the number of bytes transmitted increase from 3070 to 5390, representing a 75.5% increase when moving from Level I to Level III. From Level III to Level V, the bytes count increases from 5390 to 6396, corresponding to an 18.64% increase. Overall, the transition from Level I to Level V results in sending slightly more than twice (108.34%) the amount of data in a single handshake. This indicates that higher security levels require significantly greater data transmission, which may impact performance depending on the system constraints. The difference between using post-quantum or hybrid signatures is negligible.

9. Conclusion and Future Work

A timely transition to quantum-resistant protocols is crucial to ensuring the security of online communications. To facilitate this transition in web scenarios, this paper proposes a framework for systematically evaluating the trade-offs of integrating hybrid and post-quantum cryptographic primitives into the TLS protocol.

The proposed framework is used to evaluate TLS configurations incorporating the post-quantum schemes recently standardized by NIST, as well as new candidates from the additional signature round, demonstrating the framework's versatility in integrating new primitives with minimal effort. These configurations are analyzed under different scenarios and network conditions. Our analysis indicates that, in terms of performance, the most favorable TLS configurations are based on post-quantum KEMs, particularly at higher security levels. While these configurations incur higher data transmission costs compared to traditional KEMs, they are less costly than hybrid ones. Additionally, the main limitation to a fully post-quantum TLS is the size of post-quantum signatures. However, some of these configurations imposed an overhead not significantly higher than that of traditional TLS configurations. In addition, we analyze the cost associated with moving between security levels, considering both the reduction of connections and the increase in transmitted bytes. In the four analyzed scenarios, we observe that the least performance loss between levels occurs when fully post-quantum KEMS and signatures are involved.

The research conducted can be expanded in several ways. One area of future research involves exploring the impact of different Web PKI certification chain configurations. In this work, we consider certification chain of only two certificates, where both use the same signature algorithm. However, a typical configuration involves three certificates of various strengths. In fact, recent trends[8] emphasize the importance of using different signature algorithms at different tiers of the certification chain. Future research could evaluate various trust chain configurations to optimize security while maintaining an adequate level of performance and data exchange. Another interesting area of future research is the incorporation of post-quantum primitives in other security protocols. The Cloudflare Radar tool[9] reports that 33% of web communications use the QUIC protocol, which integrates TLS over UDP for securing web communications. Given its growing adoption, we consider the relevance of extending this study to the QUIC protocol. Such an extension would allow extrapolating our findings, improving the generalization of the results.

[8] Chromium. Building a Deployable Post-quantum Web PKI. URL: https://www.chromium.org/Home/chromium-security/post-quantum-pki-design/ [Last access on February 2025]

[9] Cloudflare Radar. URL: https://radar.cloudflare.com/adoption-and-usage [Last access on February 2025]

Acknowledgment

This work has been partially supported by ... The authors are also grateful to Jesús Rodríguez for his technical support.

References

[1] Internet Engineering Task Force (IETF). The transport layer security (tls) protocol version 1.3. https://www.rfc-editor.org/rfc/rfc8446, 2018. Accessed: 2024-09.

[2] P.W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994. doi: 10.1109/SFCS.1994.365700.

[3] Entrust. Harvest now, decrypt later: Fact or fiction, 2023. URL https://www.entrust.com/blog/2023/11/harvest-now-decrypt-later-fact-or-fiction. Accessed: 2024-10.

[4] NIST CSRC. Post-quantum cryptography. https://csrc.nist.gov/projects/post-quantum-cryptography, 2024. Accessed: 2024-07.

[5] Kris Kwiatkowski, Panos Kampanakis, Bas Westerbaan, and Douglas Stebila. Post-quantum hybrid ECDHE-MLKEM key agreement for TLSv1.3. Internet-Draft draft-kwiatkowski-tls-ecdhe-mlkem-02, 2025. URL https://datatracker.ietf.org/doc/draft-kwiatkowski-tls-ecdhe-mlkem/. Accessed: 2024-10.

[6] Michael Connolly. ML-KEM Post-Quantum Key Agreement for TLS 1.3. Internet-Draft draft-connolly-tls-mlkem-key-agreement-02, October 2024. URL https://datatracker.ietf.org/doc/draft-connolly-tls-mlkem-key-agreement/. Accessed: 2024-10.

[7] Stan Kaminsky. Where and how post-quantum cryptography is being used in 2024, 2024. URL https://www.kaspersky.com/blog/postquantum-cryptography-2024-implementation-issues/52095/. Accessed: 2024-10.

[8] Chromium Project. Protecting chrome traffic with hybrid post-quantum key agreements, 2023. URL https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html. Accessed: 2024-10.

[9] Cloudflare. Cloudflare now uses post-quantum cryptography to talk to your origin server, 2023. URL https://blog.cloudflare.com/post-quantum-to-origins/. Accessed: 2024-10.

[10] David Adrian, Emily Stark, Pavel Francírek, and Ryan Dickson. Tldr.fail, 2024. URL https://tldr.fail/. Accessed: 2024-10.

[11] Maximilian Schöffel, Frederik Lauer, Carl C. Rheinländer, and Norbert Wehn. Secure iot in the era of quantum computers —where are the bottlenecks? *Sensors*, 22(7), 2022. ISSN 1424-8220. doi: 10.3390/s22072484.

[12] Kyung-Ah Shim. On the suitability of post-quantum signature schemes for internet of things. *IEEE Internet of Things Journal*, 11 (6):10648–10665, 2024. doi: 10.1109/JIOT.2023.3327400.

[13] Maxime Buser, Rafael Dowsley, Muhammed Esgin, Clémentine Gritti, Shabnam Kasra Kermanshahi, Veronika Kuchta, Jason Legrow, Joseph Liu, Raphaël Phan, Amin Sakzad, Ron Steinfeld, and Jiangshan Yu. A survey on exotic signatures for post-quantum blockchain: Challenges and research directions. *ACM Comput. Surv.*, 55(12), March 2023. ISSN 0360-0300. doi: 10.1145/3572771. URL https://doi.org/10.1145/3572771.

[14] Zebo Yang, Haneen Alfauri, Behrooz Farkiani, Raj Jain, Roberto Di Pietro, and Aiman Erbad. A survey and comparison of post-quantum and quantum blockchains. *IEEE Communications Surveys & Tutorials*, 26(2):967–1002, 2024. doi: 10.1109/COMST.2023.3325761.

[15] Matt Braithwaite. Experimenting with post-quantum cryptography, July 2016. URL https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html. Accessed: January 29, 2025.

[16] Adam Langley. Post-quantum confidentiality for TLS, April 2018. URL https://www.imperialviolet.org/2018/04/11/pqconftls.html. Accessed: January 29, 2025.

[17] Krzysztof Kwiatkowski, Adam Langley, Nick Sullivan, Dave Levin, Alan Mislove, and Luke Valenta. Measuring tls key exchange with post-quantum kem, August 2019.

[18] Adam Langley. Real-world measurements of structured-lattices and supersingular isogenies in TLS, October 2019. URL https://www.imperialviolet.org/2019/10/30/pqsivssl.html. Accessed: January 29, 2025.

[19] Dominik Marchsreiter and Johanna Sepúlveda. Hybrid Post-Quantum enhanced TLS 1.3 on embedded devices. In *2022 25th Euromicro Conference on Digital System Design (DSD)*, pages 905–912, 2022. doi: 10.1109/DSD57027.2022.00127.

[20] Maximilian Schöffel, Frederik Lauer, Carl C. Rheinländer, and Norbert Wehn. On the energy costs of Post-Quantum KEMs in TLS-based low-power secure IoT. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, IoTDI '21, page 158–168, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383547. doi: 10.1145/3450268.3453528. URL https://doi.org/10.1145/3450268.3453528.

[21] Sebastian Paul, Felix Schick, and Jan Seedorf. Tpm-based post-quantum cryptography: A case study on quantum-resistant and mutually authenticated TLS for IoT environments. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, ARES '21, pages 1–10, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450390514. doi: 10.1145/3465481.3465747. URL https://doi.org/10.1145/3465481.3465747.

[22] Markku-Juhani O. Saarinen. Mobile energy requirements of the upcoming nist post-quantum cryptography standards. In *2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 23–30, 2020. doi: 10.1109/MobileCloud48802.2020.00012.

[23] Kevin Bürstinghaus-Steinbach, Christoph Krauß, Ruben Niederhagen, and Michael Schneider. Post-quantum TLS on embedded systems: Integrating and evaluating Kyber and SPHINCS+ with mbed TLS. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ASIA CCS '20, page 841–852, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450367509. doi: 10.1145/3320269.3384725. URL https://doi.org/10.1145/3320269.3384725.

[24] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-Quantum TLS without handshake signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, CCS '20, page 1461–1480, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370899. doi: 10.1145/3372297.3423350. URL https://doi.org/10.1145/3372297.3423350.

[25] T. Wiggers, S. Celi, P. Schwabe, D. Stebila, and N. Sullivan. KEM-based authentication for TLS 1.3. Internet-Draft, October 2024. URL https://datatracker.ietf.org/doc/draft-celi-wiggers-tls-authkem/. Expires: 20 April 2025.

[26] Douglas Stebila and Michele Mosca. Post-quantum key exchange for the internet and the open quantum safe project. In Roberto Avanzi and Howard Heys, editors, *Selected Areas in Cryptography (SAC) 2016*, volume 10532 of *LNCS*, pages 1–24. Springer, October 2017.

[27] Christian Paquin, Douglas Stebila, and Goutam Tamvada. Benchmarking post-quantum cryptography in TLS. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography*, pages 72–91, Cham, 2020. Springer International Publishing. ISBN 978-3-030-44223-1.

[28] Nouri Alnahawi, Johannes Müller, Jan Oupický, and Alexander Wiesmaier. A comprehensive survey on post-quantum TLS. *IACR Communications in Cryptology*, 1(2), 2024. ISSN 3006-5496.

[29] D. Stebila, S. Fluhrer, and S. Gueron. Hybrid key exchange in TLS 1.3. Internet-Draft draft-ietf-tls-hybrid-design-12, Internet Engineering Task Force, January 2025. URL https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/. Work in Progress.

[30] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. https://doi.org/10.6028/NIST.FIPS.203, 2024. Federal Information Processing Standards Publication (FIPS) NIST FIPS 203.

[31] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26:80–101, 2013. doi: 10.1007/s00145-011-9114-1. URL https://doi.org/10.1007/s00145-011-9114-1.

[32] National Institute of Standards and Technology. Recommendations for key-encapsulation mechanisms. https://csrc.nist.gov/pubs/sp/800/227/ipd, 2025. Draft, NIST Special Publication (SP) 800-227, accessed Jan. 27, 2025.

[33] National Institute of Standards and Technology. Module-lattice-based digital signature standard, 2024. URL https://doi.org/10.6028/NIST.FIPS.204.

[34] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru specification v1.2, 2020.

[35] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM Press, 2008. doi: 10.1145/1374376.1374407.

[36] Phillip Gajland, Jonas Janneck, and Eike Kiltz. A closer look at falcon. Cryptology ePrint Archive, Paper 2024/1769, 2024. URL https://eprint.iacr.org/2024/1769.

[37] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of ACM STOC'90*, pages 387–394, 1990.

[38] National Institute of Standards and Technology. Stateless hash-based digital signature standard, 2024. URL https://doi.org/10.6028/NIST.FIPS.205.

[39] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - a practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, pages 117–129, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[40] Jean-Philippe Aumasson, Daniel J. Bernstein, Wouter Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Szabolcs L. Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Michael M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Bas Westerbaan. SPHINCS+ – submission to the nist post-quantum project, v.3.1. https://sphincs.org/data/sphincs+-r3.1-specification.pdf, 2022. Available at the SPHINCS+ project website.

[41] Andreas Hülsing. W-OTS+ – shorter signatures for hash-based signature schemes. In Amr M. Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors, *Progress in Cryptology – AFRICACRYPT 2013*, volume 7918 of *Lecture Notes in Computer Science*, pages 173–188. Springer, 2013. doi: 10.1007/978-3-642-38553-7_10. URL https://doi.org/10.1007/978-3-642-38553-7_10.

[42] NIST. Post-Quantum Cryptography: Additional Digital Signature Schemes, 2025. URL https://csrc.nist.gov/projects/pqc-dig-sig. Accessed: 2025-02-21.

[43] Jacques Patarin. The oil and vinegar signature scheme, 1997.

[44] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 206–222, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. ISBN 978-3-540-48910-8.

[45] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Matthias J. Kannwischer. Mayo round 2 version. Submitted version, 2025. URL https://pqmayo.org/. Contact: contact@pqmayo.org.

[46] Robert McEliece. A public key cryptosystem based on algebraic coding theory. *DSN Progress Report*, 42(44):114–116, 1978.

[47] Marco Baldi, Alessandro Barenghi, Michele Battagliola, Sebastian Bitzer, Marco Gianvecchio, Patrick Karl, Felice Manganiello, Alessio Pavoni, Gerardo Pelosi, Paolo Santini, Jonas Schupp, Edoardo Signorini, Freeman Slaughter, Antonia Wachter-Zeh, and Violetta Weger. Cross: Codes and restricted objects signature scheme. Submission to the NIST Post-Quantum Cryptography Standardization Process, Version 2 - January 31, 2025, 2025. URL https://pqc-cross.org/. Algorithm Specifications and Supporting Documentation.

[48] Marco Baldi, Massimo Battaglioni, Franco Chiaraluce, Anna-Lena Horlemann, Edoardo Persichetti, Paolo Santini, and Violetta Weger. A new path to code-based signatures via identification schemes with

restricted errors. *Advances in Mathematics of Communications*, 2025.

[49] Federal Office for Information Security (BSI). Securing tomorrow, today: Transitioning to Post-Quantum cryptography, 2024. URL https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Crypto/PQC-joint-statement.html. Accessed: 2025-01-24.

[50] European Commission. Recommendation on a coordinated implementation roadmap for the transition to post-quantum cryptography, 2024. URL https://ec.europa.eu/newsroom/dae/redirection/document/104249. Accessed: 2025-02-06.

[51] Gerrit Almes, Srinivas Kalidindi, Marija Zekauskas, and Al Morton. A one-way loss metric for ip performance metrics (ippm). RFC 7680, January 2016. URL https://www.rfc-editor.org/rfc/rfc7680.txt. [Online; accessed 2025-06-19].