Towards the Quantum-Safe Web: Benchmarking Post-Quantum TLS

Ruben Rios, José A. Montenegro, Antonio Muñoz and Davide Ferraris

***Abstract*—The transition to a quantum-resistant Internet is a complex process that depends on the integration of post-quantum cryptographic primitives into existing security protocols. This paper analyzes the impact that the primitives selected by NIST in their post-quantum cryptography competition has on a critical Internet security protocol, the TLS protocol. The analysis is facilitated by a framework that enables the implementation of an evaluation scenario in which different post-quantum primitives can be tested under identical conditions, ensuring a fair comparison. Our results indicate that the computational overhead introduced by current post-quantum standards in TLS is comparable to that of traditional algorithms, and even more efficient at high security levels. However, their significant impact on data transmission constrains the transition to a full-fledged quantum-safe web.**

***Index Terms*—Cybersecurity, Transport Layer Security, Post-Quantum Cryptography.**

I. Introduction

QUANTUM computing has been touted as the next great technological revolution, with major companies investing heavily in its development. Although it is still under development, its huge potential can already be envisaged in applications such as industrial process optimization, artificial intelligence and the design of new drugs and materials. Quantum computing will also revolutionize cybersecurity.

The principles of quantum physics will enable quantum computers to create virtually unbreakable communication systems, thanks to mechanisms such as quantum key distribution, which can detect any attempt to intercept or modify the quantum particles used for the exchange. The randomness of quantum systems will also make it possible to generate high-quality random numbers, which are essential for cryptography. However, the downside of quantum systems is that they will also be able to drastically reduce the level of security of today's cryptography, thus undermining the foundations of Internet security [1].

In response to this daunting threat, the research community is developing new cryptographic primitives that can withstand quantum attacks. This area of research has been referred to as post-quantum cryptography (PQC), but also as quantum-safe, or quantum-resistant cryptography. In fact, the US National Institute of Standards and Technology (NIST) initiated a competition in 2016 to standardize PQC algorithms [2]. After over 80 initial submissions and three rounds of competition submissions, four algorithms were selected for standardisation.

Despite the availability of PQC standard primitives, integrating them into current systems and networks requires a substantial amount of research and engineering [3]. These primitives encompass a considerable number of configuration options, each of which not only has security implications but also impacts the performance of systems and protocols, being particularly relevant the case of the Transport Layer Security (TLS) protocol, which is the security backbone of the world's online activities, including web browsing, e-commerce and numerous other applications.

While some progress has been made in integrating PQC into TLS, these efforts are still in the early stages. For example, the IETF is drafting new standards [4], and several major browsers have started offering partial support for PQC, however, only a small fraction of clients currently use it, and compatibility issues exist. These initial steps would greatly benefit from tools and frameworks that enables the assessment and comparison of TLS configurations involving PQC primitives before implementing them in real settings.

This is precisely one of the primary goals of this paper, to provide a framework for evaluating the impact of integrating PQC primitives into TLS. Although some efforts have been made in this direction [5], our framework provides flexible scenarios that can be adapted to different network conditions, and can be easily extended with new PQC primitives as they are developed. In fact, it has been used to evaluate the performance and transmission overhead of TLS handshakes incorporating the latest PQC primitives, including NIST standards.

II. Traditional Web Security

TLS is a fundamental security protocol of the Internet. Whenever someone visits a website, prior to the transmission of any data, a secure connection is established to ensure that third parties are unable to eavesdrop, tamper with, or forge messages. Consequently, the authenticity, confidentiality, and integrity of web-based communications are guaranteed. The most recent version of the protocol is TLS 1.3 (defined in RFC 8446), but TLS 1.2 is still in use. All previous versions have been deprecated.

Generally speaking, TLS consists of two parts: handshake and data transmission. Fig. 1 illustrates the message flow in TLS 1.3, with the dashed boxes representing messages that are not always sent. The handshake can be divided in three phases: key exchange, server parameters and authentication. In the *key exchange* phase, which includes the `ClientHello` and `ServerHello` messages, the client and the server negotiate the protocol version, determine appropriate cryptographic

This paper was produced by the Network, Information and Computer Security (NICS) lab from Universidad de Málaga, Spain.

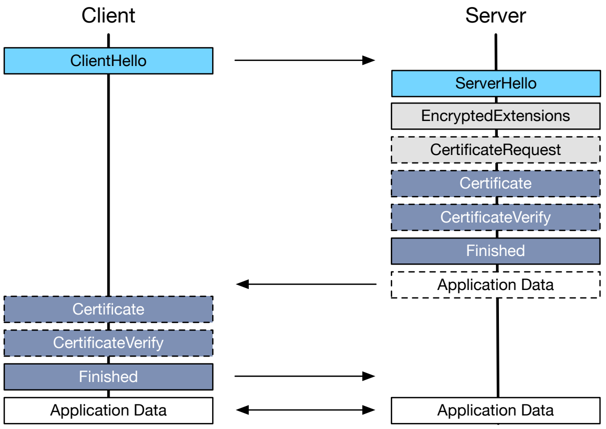


Fig. 1: TLS 1.3 Message Flow (based on RFC 8446)

parameters and establish shared secret keys. All handshake message after the key exchange phase will be encrypted using the agreed keys.

The `ClientHello` message contains a list of supported algorithms in descending order of client preference: Ephemeral Diffie-Hellman (DHE) or Pre-Shared Key (PSK). DHE can be used over finite fields (FFDHE) or over elliptic curves (ECDHE), being the latter usually preferable as it can achieve the same security guarantees with significantly shorter key shares (see TABLE I). The elliptic curves used in TLS 1.3 are secp256r1, secp384r1, and secp521r1 (or NIST P-256, P-384, and P-521) as well as x25519 and x448. Note that a unique key share value for each of the listed algorithms can be added to the message. In contrast, the `ServerHello` message contains only a key share for the algorithm selected by the server from those offered by the client. In scenarios where a secret key is pre-established, no values are shared between the client and the server.

The *server parameters* phase comprises two messages, `EncryptedExtensions` and `CertificateRequest`, the latter being sent only in case the server requires client authentication. During the *authentication phase*, the client may authenticate the server (server-only) or they may mutually authenticate by exchanging three types of messages. The `Certificate` message typically contains the certificate of the endpoint and `CertificateVerify` contains a signature over the entire handshake using the private key associated with the certificate in the previous message. These messages will only be present if authentication is done via certificates and not by means of a pre-shared key. The signature algorithms used for authentication are either RSA, Elliptic Curve Digital Signature Algorithm (ECDSA) or the Edwards-Curve Digital Signature Algorithm (EdDSA). The elliptic curves used in both cases are the same as those used during the key exchange phase.

The `Finished` message is always present and serves to confirm the secret key and identity of the other endpoint. At this point, the client and server derive the keying material necessary to exchange application-level data protected with authenticated encryption.

III. Security Levels

The security level or strength of cryptographic primitives is typically expressed as *number of bits*. A primitive with n bits of security requires an effort equivalent to 2^n runs of the most effective attack to break it. For example, the security of AES is based on the need to brute-force the key space, while the security of RSA is based on the execution complexity of the best known algorithm for factoring large primes. This is why two primitives may have comparable strength despite having different key sizes.

TABLE I: Security strength comparison
(based on NIST SP 800-57 Part 1 Rev. 5)

Security strength	Symmetric Key Cryptography	Public Key Cryptography	
		Finite Fields	**Elliptic Curves**
<80	2TDEA	1024	160-223
112	3TDEA	2048	224-255
128	AES-128	3072	256-383
192	AES-192	7680	384-512
256	AES-256	15360	≥512

The equivalence shown in TABLE I provides an accurate framework for comparing different types of cryptosystems. However, in view of the uncertainties associated with post-quantum primitives, including the potential for new quantum attacks and the difficulty of predicting the performance of quantum computers, NIST established a new framework comprising broader security categories. These categories are based on estimations of the necessary resources to successfully attack primitives which are deemed to be reasonably robust against quantum attacks, namely AES and SHA3. Five security categories or levels are considered [2]:

- **Level I**: key search on AES-128.
- **Level II**: collision search on SHA3-256.
- **Level III**: key search on AES-192.
- **Level IV**: collision search on SHA3-384.
- **Level V**: key search on AES-256.

The computational resources needed to launch these attacks can be quantified in terms of circuit complexity or circuit depth as running extremely long serial computations in quantum computers is a extremely challenging process. Consequently, a post-quantum primitive belongs to security level I if the computational resources required to break it are comparable to those required to perform a key search attack against AES 128-bit using the best known attack, which in this case is Grover's quantum algorithm [6]. Although Grover's algorithm achieves a quadratic speedup over a classical brute-force attack, the complexity of the attack lies in the implementation of the quantum circuit needed to run it.

Participants to the post-quantum standarization contest organized by NIST were asked to classify their algorithms according to this framework.

IV. Post-Quantum Standards

The first quantum algorithms with the potential to compromise the security of current cryptographic primitives were devised in the mid-1990s. One particularly noteworthy example is Shor's algorithm [7]. The algorithm is capable of performing the prime factorisation of large integers and solving

TABLE II: Security Level of Selected PQC Algorithms

		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

discrete logarithms, which serve as the foundation of public-key cryptosystems, in polynomial time. However, it was not until recent years, with the recent technological developments in quantum computing, that the necessity for standardisation of quantum-resistant algorithms became apparent.

Despite the existence of several standardization initiatives (e.g., [8], [9]), the most notable effort has been led by NIST [2]. In 2016, the agency launched a worldwide competition with the aim of evaluating and standardizing PQC algorithms that could replace current public-key cryptosystems as they are particularly vulnerable to Shor's algorithm.

Following the initial submission of 69 algorithms and three rounds of competition, which led to the discovery of vulnerabilities and the refinement of proposals [10], the first PQC algorithms selected for standardisation were announced by NIST in July 2022. The selection included:

- **CRYSTALS-Kyber** for key encapsulation and public-key encryption.
- **CRYSTALS-Dilithium**, **Falcon**, and **SPHINCS$^+$** for digital signatures.

With the exception of SPHINCS$^+$, which is based on cryptographic hash functions, all of the aforementioned algorithms are based on problems over mathematical structures called lattices. Although lattice-based algorithms were the most efficient ones, the hash-based algorithm was included as a safeguard mechanism. Following the same rationale, a fourth round of the competition started with algorithms based on different problems, including elliptic curve isogenies and error-correction codes.

The algorithms submitted for evaluation were required to include an security strength estimate for each parameter set according to the categories defined by NIST (see Section III), as shown in TABLE II. In the particular case of SPHINCS$^+$, the authors submitted two optimized versions for each security level: one optimized in size (ending on 's' for small) and one optimized in performance (ending on 'f' for fast). Moreover, each version of the algorithm can be instantiated with three different hash functions: SHA-256, SHAKE256 and Haraka. On top of that, there are simple and robust instantiations, where the difference lies in the input to the hash functions. This results in numerous signature schemes.

In the summer of 2023, three Federal Information Processing Standards (FIPS) drafts were published for use by all non-military US government agencies and government contractors, but they also have international relevance [11]. These drafts are based on the selected algorithms with minor changes [2]:

- **FIPS 203**: *Module-Lattice-Based Key-Encapsulation Mechanism Standard* (ML-KEM), based on Kyber.
- **FIPS 204**: *Module-Lattice-Based Digital Signature Standard* (ML-DSA), based on CRYSTALS-Dilithium.
- **FIPS 205**: *Stateless Hash-Based Digital Signature Standard* (SLH-DSA), based on SPHINCS$^+$.

One year later, during the summer of 2024, these drafts were finally designated as FIPS standards. Currently, NIST is also working on the development of a FIPS for the standardisation of a digital signature algorithm derived from Falcon. New FIPS standards may eventually emerge from the fourth round of the competition, which focuses on digital signatures.

V. Quantum-Safe Web

As of 2021, over 90% of web traffic used HTTPS, which is secured by TLS [12]. This figure is corroborated by the Google Transparency Report, which indicates that 96% of their products are accessed via HTTPS in 2024 [13]. However, the cryptographic algorithms currently used in TLS during the handshake phase (mainly Level I Curve 25519 and RSA) are vulnerable to quantum attacks.

The transition towards a quantum-safe web is not without challenges. The process involves the integration of PQC primitives into the TLS protocol while ensuring compatibility, and maintaining performance efficiency. In early 2024, only a small fraction (below 2%) of TLS clients supports post-quantum key agreements, but implementing post-quantum signatures is considered to be much more challenging, and will thus require more time [14].

A. Evaluation Framework

The core component of our evaluation framework is OpenSSL[1]. OpenSSL is a widely used open-source library that provides all the necessary algorithms and tools for securing communications over networks, such as the generation of the public and private keys and their corresponding certificates used during the TLS protocol handshake. Furthermore, OpenSSL provides three commands for testing TLS connections. The `s_server` command is responsible for creating a TLS server, while `s_client` and `s_time` commands are used for the establishment of connections with the server and measuring the performance of TLS connections. The version of the OpenSSL library used is 3.3.1, which is the latest stable version at the time of writing.

[1] https://github.com/openssl/openssl

Another key component of our framework is the `liboqs`[2] library. This library is part of the Open Quantum Safe (OQS) project, whose primary goal is to develop quantum-safe cryptography for real-world applications. It provides us with the implementation of PQC primitives.

The final component is the `oqsprovider`[3], which enables the integration of the PQC algorithms available in `liboqs` into OpenSSL. It should be noted that not all PQC standards are currently available in this component. For example, there is still no implementation of `SLH-DSA`, although an earlier version, Sphincs$^+$, is available. However, these standards could be easily integrated into our framework as soon as they are implemented.

In order to facilitate the creation of an environment for testing all the cryptographic primitives under the same conditions, a Docker image has been created that includes the core elements of our framework, as well as all the scripts for the execution of our tests.

A particular script can be initiated via Docker, specifying the cryptographic algorithms to be employed by TLS for key establishment and authentication.

```
docker run  -e KEM_ALG=<kem_selected>
            -e SIG_ALG=<sign_selected>
            -it oqs-curlv331 <script_selected>
```

Although in this paper we concentrate on an ideal scenario, the proposal can be adapted to more realistic scenarios by using environment variables representing packet delay and packet loss, (`TC_DELAY` and `TC_LOSS`, respectively) that the script use to modify its behavior.

```
docker run  --cap-add=NET_ADMIN
            -e TC_DELAY=<delay>
            -e TC_LOSS=<loss>
            -e KEM_ALG=<kem_selected>
            -e SIG_ALG=<sign_selected>
            -it oqs-curlv331 <script_selected>
```

Internally, the scripts use the aforementioned OpenSSL commands. On the server side, the command used to measure the number of connections established per second and the quantity of data transmitted is OpenSSL's `s_server`. The following line generates a TLS 1.3 server on port 4433 with the certificates created for the server:

```
openssl s_server -cert <server_crt> -key <server_key>
                 -www -tls1_3
                 -accept <server_ip>:<server_port>
```

To determine the number of successful connections, the client uses the `s_time` command to establish TLS connections to the previously created server. A time parameter is used to indicate to the client the number of seconds it should initiate new connections with the server. We use this parameter to compare the performance of TLS when using different cryptographic primitives. In our tests, a 10-second interval was selected.

```
openssl s_time -connect <server_ip>:<server_port>
               -new -time <exec_time> -verify 1
               -CAfile <CA_crt>
```

Although the `s_time` command establishes multiple connections, we decided to run it 100 times to perform a statistically significant analysis of the results.

Finally, to evaluate the transmission overhead, our scripts use the `s_client` command, which is employed to establish a connection with the server. This command provides information about the number of kilobytes transmitted during the handshake. The transmission overhead is primarily influenced by the size of the certificates being exchanged, but also the secret shares used during the key exchange phase.

```
openssl s_client -connect <server_ip>:<server_port>
                 -state -servername <cname>
                 -CAfile <CA_crt> -showcerts
```

The source code required to replicate our tests, along with the corresponding results, is available for download from our public GitHub repository[4]. The repository also includes the results of executions conducted under different network conditions.

B. *Handshake Performance Impact*

This section examines the impact that selecting traditional or post-quantum algorithms has on TLS performance, measured by the number of successfully completed handshakes over a ten-second period. The results are shown in Fig. 2, where different TLS configurations are compared. Each configuration consists of a KEM algorithm along with one of the signature algorithms listed on the x axis. When available, we evaluate the standard PQC algorithm, rather than the finalist from the NIST competition[5]. The figure uses box plots to highlight the most representative values of the distribution, including the median, quartiles and outliers, represented by the + symbol.

At Level I (Fig. 2a), the choice of KEM algorithms does not significantly impact the number of connections established. Using the traditional P-256 as KEM exhibits slightly lower performance (below 20%) than X25519 and the post-quantum ML-KEM512. With respect to post-quantum signatures, Falcon would be the best option as it offering a performance that lies between the traditional RSA and Ed25519. On the other hand, a transition based on SPHINCS$^+$ would have a significant impact on performance compared to Falcon. In particular, the fast and small variants of SPHINCS$^+$ reduce their capacity to establish connections by a factor of approximately 10 and 140, respectively. The boxplot shows that the distribution of results exhibit a relatively low dispersion, with RSA and SPHINCS$^+$ being the most stable distributions although some outliers exist for all TLS configurations.

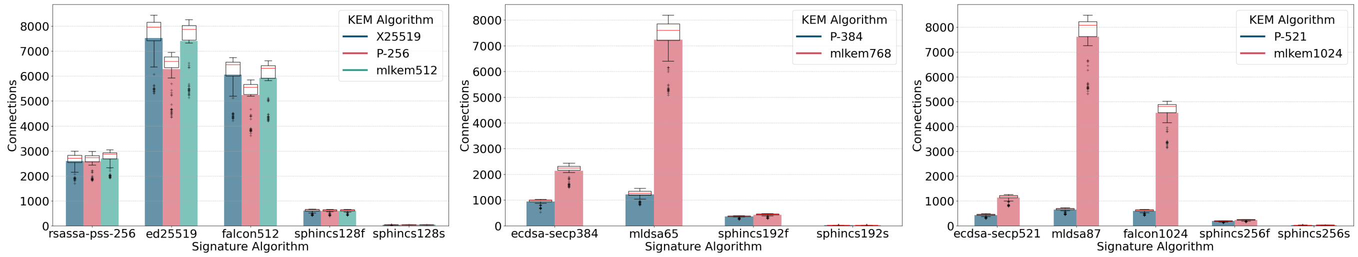
At Level III (Fig. 2b), the choice of ML-KEM leads to a significant enhancement in performance when used in conjunction with a signature algorithm based on traditional (P-384) or post-quantum (ML-DSA). This combination of post-quantum algorithms results in a six-fold performance increase compared to traditional algorithms. Again, the selection of SPHINCS$^+$ is not recommended at Level III. The fast variant exhibits a performance reduction by a factor of 17, while the small variant is around 286 times less efficient, with an

[2] https://github.com/open-quantum-safe/liboqs

[3] https://github.com/open-quantum-safe/oqs-provider

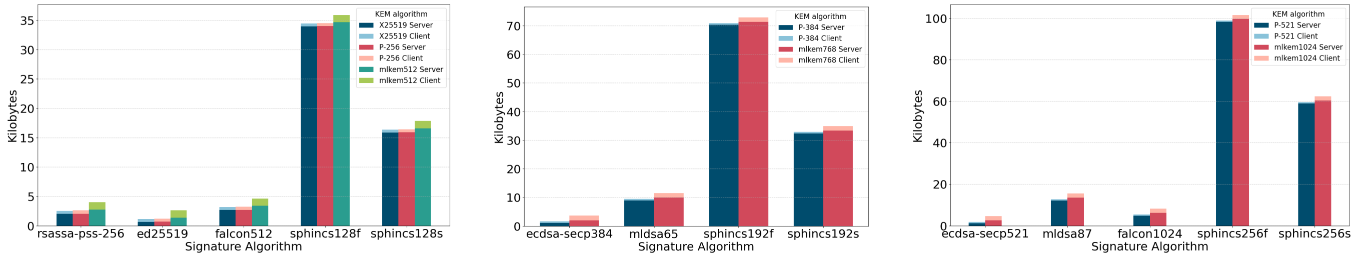
[4] https://github.com/montenegro-montes/TLS-PQ

[5] According to our tests the standard versions perform slightly better.



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

Fig. 2: TLS Handshake connections during 10 seconds



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

Fig. 3: Data Transmission in Kilobytes During TLS handshake

average of 25 connections over 10 seconds. According to the boxplot, most executions present similar performance but there are still some atypical executions where the performance is notably lower. This is most evident in configurations with higher average performance.

At Level V (Fig. 2c), the choice of ML-DSA as the signature algorithm ensures optimal performance, irrespective of whether a traditional (P-521) or post-quantum (ML-KEM1024) key exchange algorithm is employed, outperforming the FALCON algorithm. Finally, if SPHINCS$^+$ fast and small variants are considered, their performance is respectively 33 and 272 times inferior than the optimal choice of ML-KEM and ML-DSA. In terms of dispersion, the results are consistent with lower security levels.

In conclusion, at all security levels, there is a well-performing fully post-quantum TLS configuration (i.e., based solely on PQC). Interestingly, traditional TLS configurations (based on elliptic curves and RSA) only perform better than post-quantum TLS configurations at Level I, when Curve 25519 is used for authentication, but not for RSA. The best-performing TLS configurations at Levels III and V are fully post-quantum. However, the use of either SPHINCS$^+$ variant consistently results in significant performance degradation.

C. Data Transmission Impact

This section explores the number of kilobytes transmitted during a TLS handshake when combining different algorithms for key exchange and for authentication, including both classical and post-quantum primitives. The results, shown in Fig. 3, are organized based on the security level of the algorithms and show client-side and server-side transmissions separately. We concentrate on server authentication as it represents the most common scenario in web communications.

As shown in Fig. 3, the total number of kilobytes transferred during a full handshake is consistently lower when classical primitives are employed in comparison to post-quantum primitives. The use of ML-KEM for key exchange results in larger data transmissions in comparison to classical primitives. At Level I, the increase in data transmission is approximately 155% when it replaces RSA and up to 227% when it is used instead of X25519. At Level III, the increase remains consistent with the previous level. At Level V, when replacing P-521 by ML-KEM, the transmission overhead increases by a factor of 2.5, with a handshake that amounts to a total of approximately 4700 kilobytes.

On the other hand, the transmission overhead of post-quantum signature algorithms is significantly higher, being FALCON the most affordable solution. At Level I (see Fig. 3a), the combination of ML-KEM and FALCON yields an increase in data transmission around 177% in comparison to the combination of P-256 and RSA, and a four-fold increase with respect to X25519 and ED25519. In contrast, SPHINCS$^+$ presents the highest overhead for all security levels, even for the size-optimized variant, which involves approximately half the overhead of the performance-optimized version. In the worst case scenario, SPHINCS$^+$ transfers over 100 kilobytes during a single TLS handshake (see Fig. 3c) compared to approximately 1.5 kilobytes for classical primitives. ML-DSA lies in between, being approximately 5.7 times worse than a classical handshake with curve P-384 (Fig. 3b) and up to 7 times worse for the highest security level (Fig. 3c).

In summary, traditional TLS configurations impose a relatively low overhead compared to post-quantum TLS. In more detail, the transmission impact of post-quantum KEM primitives is not significant, whereas the associated costs for signatures are substantially higher. The most favorable post-quantum combination is the use of ML-KEM and FALCON, which yields results comparable to classical primitives. Conversely, SPHINCS$^+$ is extremely costly irrespective of the security level. In general, the volume of data transmitted grows with higher security levels.

VI. DISCUSSION

The evaluation of TLS configurations presented in the previous section focuses on handshake performance and volume of data transmitted under ideal network conditions. A key consideration is that while some algorithms may excel in performance, they often result in significantly larger data transmissions, which can congest communication channels and reduce network efficiency. Therefore, it is essential to consider both parameters together, as balancing performance with transmission efficiency is critical.

At Level I, the performance of FALCON is superior to that of RSA, although it does not reach the level of ED25519. Furthermore, there is no discernible impact on the volume of data transmitted during TLS handshakes. In contrast, the selection of SPHINCS$^+$ has a substantial impact on both performance and the volume of kilobytes transmitted. Moreover, the combination of FALCON and ML-KEM can provide a secure and efficient solution for transitioning from traditional to post-quantum TLS at Level I, as ML-KEM has a performance similar to that of currently used key exchange mechanisms.

At Level III, there is no FALCON implementation, making ML-DSA the most advantageous post-quantum signature option. Combined with ML-KEM, significantly improves performance over traditional algorithms. The downside of this choice is that the information transfer is almost 6 times higher than the traditional one. In the case of SPHINCS$^+$, its performance is notably inferior to that of traditional algorithms. Moreover, it imposes a significant data transmission overhead, rendering it unsuitable for TLS. Regarding the KEM algorithm, the transition is guaranteed by choosing ML-KEM. This is one of the reasons why the IETF is currently working on standardizing the inclusion of ML-KEM in TLS, with preliminary support already implemented in some web browsers.

The performance of ML-KEM remains reasonable at Level V. Regarding signature algorithms, both ML-DSA and FALCON outperform traditional schemes; however, it is worth highlighting the outstanding improvement demonstrated by ML-DSA. When evaluating the volume of data transmitted, FALCON is the preferred choice as it only introduces a slight increase in transmission compared to traditional algorithms, significantly lower than the overhead introduced by ML-DSA. The choice of the SPHINCS$^+$ algorithm is discouraged due to its inadequate performance and the increase in data transmission by a factor of more than 50.

In summary, the use of ML-KEM in TLS is highly recommended as it introduces performance benefits at all security levels, although it comes with a reasonable increase in the volume of data transmitted. With regard to post-quantum signatures, the optimal choice at Level I is the FALCON algorithm, offering approximately twice the performance of RSA and nearing the efficiency of ED25519. This advantage becomes particularly evident when paired with ML-KEM, resulting in only a modest increase in the volume of data transmitted. The ML-DSA algorithm also brings a significant performance improvement at Levels III and V, but this comes at the cost of a substantial increase in data transmission, rendering its use inadvisable for TLS. The communication overhead observed in our tests is primarily attributed to the authentication phase, which is precluding the use of post-quantum signatures in TLS. Some initiatives [15] are underway to address this issue and facilitate a full transition to post-quantum TLS.

When packet delays and losses are introduced, the performance of TLS decreases significantly for all configurations. Our internal tests[6] show that the performance, measured in terms of the number of connections, degrades as the delay grows, and all TLS configurations tend to the same performance since the computational cost becomes negligible compared to the delay. In contrast, configurations using SPHINCS$^+$ are not notably affected by the delay because their bottleneck is on the execution of the algorithm primitives. The same behavior is observed as the packet loss rate increases, but it also impacts the volume of data transmitted.

Consequently, in the presence of packet loss and delay, the choice of the algorithm is not so crucial in terms of performance. However, it is even more critical in terms of transmission overhead, especially for configurations using SPHINCS$^+$ signatures and higher security level configurations.

VII. CONCLUSION

The imminent threat of quantum computers is driving the need to transition to an Internet secured by post-quantum technologies. While post-quantum cryptographic primitives are already available, a successful transition to a quantum-safe Internet requires a thorough understanding of how integrating these primitives affects existing secure communications protocols.

This paper focuses on analyzing the impact of state-of-the-art post-quantum key exchange and signature mechanisms on the TLS protocol. To this end, a comprehensive evaluation framework is proposed, allowing different TLS configurations to be tested under identical conditions. This framework is highly valuable for making informed decisions on the use of traditional or post-quantum cryptographic primitives in TLS, as it simplifies the selection, testing and comparison of various primitives.

Our results indicate that the transition from traditional to post-quantum KEM primitives is assured with the ML-KEM algorithm, as it matches the performance of traditional algorithms at Level I, and significantly enhances it at Levels III and V, despite an affordable increase in data transfer. For the signature algorithms, FALCON would be our preferred choice for a transition at Level I and ML-DSA at Levels III and V, although it is important to consider the traffic increase associated with this option. Finally, SPHINCS$^+$ is discouraged due to its performance limitations and the excessive amount of traffic it requires.

[6] Results can be found in our GitHub repository.

Future work will focus on several key areas to further advance this field. One critical direction involves evaluating the energy consumption of post-quantum algorithms, which is particularly relevant in resource-constraint scenarios. In addition, we plan to further investigate more realistic scenarios that account for factors such as packet loss and delay, as this will provide valuable insights into the performance and reliability of post-quantum TLS implementations. Similarly, scenarios involving mutual authentication deserves attention to ensure robust two-way authentication in quantum-safe environments. Finally, the evaluation hybrid schemes, in which traditional and post-quantum primitives coexist for key exchange and digital signatures, is another area for further exploration as this approach is expected to enable a more secure transition to a quantum-safe Internet.

Acknowledgment

This work has been partially supported by project SecTwin 5.0 (TED2021-129830B-I00), funded by MCIN/AEI/10.13039/501100011033 and the European Union ”NextGenerationEU”/PRTR. Funding for open access charge: Universidad de Málaga / CBUA.

References

[1] A. Kramer, “Quantum algorithm offers faster way to hack internet encryption,” *Science*, vol. 381, no. 6664, pp. 1270 – 1270, 2023.

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

[3] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, “Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH,” in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 2020, pp. 149–156.

[4] Internet Engineering Task Force (IETF), “Post-quantum hybrid ECDHE-MLKEM key agreement for TLSv1.3,” https://datatracker.ietf.org/doc/draft-kwiatkowski-tls-ecdhe-mlkem/, Accessed: 2024-10.

[5] C. Paquin, D. Stebila, and G. Tamvada, “Benchmarking post-quantum cryptography in TLS,” in *Post-Quantum Cryptography: 11th International Conference, PQCrypto 2020, Paris, France, April 15–17, 2020, Proceedings 11*. Springer, 2020, pp. 72–91.

[6] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC ’96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219.

[7] P. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[8] European Telecommunications Standards Institute (ETSI), “Quantum-Safe Cryptography (QSC),” https://www.etsi.org/technologies/quantum-safe-cryptography, Accessed: 2024-10.

[9] Internet Engineering Task Force (IETF), “Post-Quantum Use In Protocols,” https://datatracker.ietf.org/wg/pquip/about/, Accessed: 2024-10.

[10] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. M. Kelsey, J. Lichtinger, Y.-K. Liu, C. A. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone, “Status report on the third round of the NIST Post-Quantum cryptography standardization process,” National Institute of Standards and Technology, Tech. Rep. NIST IR 8413-upd1, 2022.

[11] Post-Quantum Cryptography Coalition, “International PQC Requirements,” https://pqcc.org/international-pqc-requirements/, August 28 2024, Accessed: 2024-10.

[12] Electronic Frontier Foundation, “Encrypting the Web,” https://www.eff.org/encrypt-the-web, Accessed: 2024-07.

[13] Google, “HTTPS encryption on the web,” https://transparencyreport.google.com/https/, Accessed: 2024-07.

[14] The Cloudfare Blog, “The state of the post-quantum Internet,” https://blog.cloudflare.com/pq-2024, Accessed: 2024-07.

[15] P. Schwabe, D. Stebila, and T. Wiggers, “Post-quantum TLS without handshake signatures,” in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1461–1480. [Online]. Available: https://thomwiggers.nl/publication/kemtls/

VIII. Biography Section

Ruben Rios (ruben.rdp@uma.es) is Associate Professor and member of NICS Lab at the University of Malaga, Spain. His main research interests are privacy enhancing technologies and applied cryptography. ORCID: https://orcid.org/0000-0002-6251-4897

José A. Montenegro (jmmontes@uma.es) is Associate Professor in the Computer Science Department at the University of Málaga, Spain, since 2010 and a member of NICS Lab at the University of Malaga. His main research interests are in artificial intelligence and applied cryptography. ORCID: https://orcid.org/0000-0001-6967-0801

Antonio Muñoz (anto@uma.es) is an Assistant Professor and a member of NICS Lab at the University of Malaga. Antonio’s research focuses on software-based attacks and defenses on microarchitectural layers in hardware and software, hardware-based secure system design and security communications. ORCID: https://orcid.org/0000-0002-6751-0625

Davide Ferraris (ferraris@uma.es) is an Assistant Professor and a member of NICS Lab at the University of Malaga. Davide’s research focuses on trust computation, its relation with security and privacy and their integration in the Internet of Things (IoT) and the Social Internet of Things (SIoT). ORCID: https://orcid.org/0000-0003-3035-3774