Integrating Post-Quantum Cryptography Within Existing Public Key Infrastructures for Complex Corporate Ecosystems

Davide Varcasia[1,2,*,†], Michele Amoretti[2]

[1]Crédit Agricole Italia, Via La Spezia 138/A, 43126, Parma, Italia
[2]University of Parma, Parco Area delle Scienze 181a, 43124, Parma, Italy

Abstract

Recent events in the world of quantum cryptography have demonstrated the need for evolution and alignment with what will soon become an evolutionary standard for all companies involved in the IT world. The hope is that in the coming years, by 2029, defined as the entry point to the Quantum Era, all participating companies will have to adopt Quantum Safe solutions, ensuring the security of communications and data when quantum technologies begin to be widely used.

This paper aims to suggest and demonstrate a practical approach, from an enterprise perspective, using technology already developed and available on the market to migrate to post-quantum cryptography. It highlights the key points to focus on: defining an appropriate network architecture, integrating it into a pre-existing global enterprise architecture, and gradually converging towards a fully post-quantum solution.

Keywords

Post-Quantum Cryptography, PKI, Certification Authority, CA Workflow

1. Introduction

The emergence of quantum computing technology may undermine numerous existing cryptographic techniques, particularly public-key cryptography, which is widely adopted to safeguard digital information. Most algorithms on which we rely are utilized globally in various communication, processing, and storage systems. All public-key algorithms and related protocols will be susceptible to hackers, rivals, and other adversaries once practical quantum computers become available. In fact, there are quantum algorithms that can break the most widely adopted public-key algorithms. The most famous example is Shor's algorithm [1] for factoring integers in polynomial time, which poses a serious threat to RSA. To protect data from future attacks, it is imperative to start preparing for the replacement of these cryptosystems with quantum-resistant ones [2].

In 2022, the National Institute of Standards and Technology (NIST) launched a competition to select new post-quantum cryptography (PQC) algorithms to be set as standards. In 2024, three quantum-resistant algorithms were selected and published. The first standard is FIPS 203 [3], based on the CRYSTALS-Kyber algorithm, which has been renamed ML-KEM, short for Module-Lattice-Based Key-Encapsulation Mechanism. The second standard is FIPS 204 [4], which is intended to serve as the primary standard for protecting digital signatures. The standard uses the CRYSTALS-Dilithium algorithm, which has been renamed ML-DSA, short for Module-Lattice-Based Digital Signature Algorithm. The third standard is FIPS 205 [5], which is also designed for digital signatures. The standard employs the Sphincs+ algorithm, which has been renamed SLH-DSA, short for Stateless Hash-Based Digital Signature Algorithm. The standard is based on a different mathematical approach from ML-DSA and is intended as a backup method in case ML-DSA proves vulnerable. A fourth standard, denoted FIPS 206, is still in the process of being defined.

Joint National Conference on Cybersecurity (ITASEC & SERICS 2026), February 09-13, 2026, Cagliari, IT
*Corresponding author.
†These authors contributed equally.
davide.varcasia@unipr.it (D. Varcasia); michele.amoretti@unipr.it (M. Amoretti)
0009-0006-1497-0878 (D. Varcasia); 0000-0002-6046-1904 (M. Amoretti)

A major challenge addressed in this work is integrating these standards within existing Public Key Infrastructures (PKIs), which are systems for distributing digital certificates in order to authenticate subjects over a network. The discussion around transition timelines, leading strategies to protect systems against quantum attacks, and approaches for combining pre-quantum cryptography with PQC to minimize transition risks started a few years ago [6], but only a few examples of architectures have emerged so far [7, 8]. Recently, the Internet Engineering Task Force (IETF) has produced a supporting document that specifies the use of NIST-standardized signature algorithms within the Public Key Infrastructure [9]. This is probably the most significant progress made so far.

In this paper, a practical strategy is presented for integrating PQC algorithms into PKIs that are part of complex corporate ecosystems, suggesting software tools to integrate Post-Quantum algorithms and procedure integration into already existing network infrastructure. The main idea is to integrate this type of algorithm into an already consolidated and complex corporate infrastructure, using digital certificates that include both the NIST-standardized algorithms, whether for signatures or symmetric key generation, as well as the classic algorithms currently used in applications.

Given the logistical and management complexity of a large enterprise, which involves multiple teams managing various technologies, the problem must be approached in such a way that the use and subsequent migration to these new algorithms must be as synergistic as possible with the entire corporate ecosystem. In this particular case, the best approach was to use hybrid digital certificates, i.e., certificates that include two distinct algorithms capable of coexisting simultaneously. This approach allows the type of algorithm to be changed based on the maturity of the technology being used. The proposed architecture was implemented as a prototype, allowing the simulation of the workflow in a portable environment.

The paper is organized as follows. In Section 2, related work on PQC integration is discussed. In Section 3, the proposed PQC-enhanced PKI architecture is described. In Section 4, the implemented prototype is presented. In Section 5, the performance evaluation of the prototype is illustrated and discussed. Finally, in Section 6, the key insights are summarized and a outline for future work is provided.

2. Related Work

The integration of PQC algorithms within existing PKIs is a challenging problem [10, 11], mainly due to the complexity of corporate ecosystems and the different timelines adopted by stakeholders. Some relevant works from the literature are summarized and discussed in this section.

Bene et al. [7] showed the importance of integrating PQC algorithms within PKIs, providing an overview of the possible algorithms that can be used and their levels of security based on the various parameterizations chosen. The authors present a performance analysis of quantum-safe solutions in use at the time of writing and, in comparison, propose new, well-performing solutions for a quantum-resistant PKI. The proposed methodology is sound, but the PQC algorithms considered are pre-NIST standardization.

Di Santo et al. [8] proposed using a hybrid approach between PQC algorithms and Quantum Key Distribution (QKD). This approach has an architecture that implements a Crypto-Machine, which includes two distinct modules, namely a Post-Quantum module and a QKD module, to ensure crypto-agility, i.e., to guaranty functional continuity based on temporary unavailability. The Crypto-Machine works as follows: first, an attempt is made to establish a key between the parties using the QKD module. If the communication channel is too noisy, a rollback strategy is applied, leveraging the Post-Quantum module. The article develops and illustrates the logical models of protocols for managing communications between two entities, including the case of switching between the two modules. A practical implementation is lacking, but the work deserves consideration.

Henrich et al. [12] show that prior to selecting a PQC scheme replacement for TLS, it is important to perform an analysis of the anticipated network conditions for applications that require a high level of responsiveness. In a more recent work [13], the same authors complement previous experiments

to include digital signature PQC schemes and hybrid variants, as well as various compositions of certificate chains. The authors conclude that migrating TLS to PQC-only or hybrid usage can generally be undertaken with a high degree of confidence. However, they recommend a cautious transition, considering suboptimal network conditions or the use of higher security levels. In such cases, the configuration of certificate chains or increasing the Transmission Control Protocol (TCP) Congestion Window (CW) might prove beneficial.

Mehmood A. and Tuveri N. [14] proposed an approach, as done in this work, focusing on the crypto-agility of the tools that will be used in the migration to Post-Quantum algorithms. Specifically, they proposed the concept of 'shallow' Provider in OpenSSL and an instance denoted as 'aurora,' which enables calling external algorithms within the OpenSSL command line. In their work, the authors proposed a logical implementation scheme and a repository where their module can be used on GitHub. What is presented in this article represents an excellent evolutionary step for the OpenSSL tool, as it provides customization based on operational needs. Since OpenSSL is purely command-line software, we opted to add a tool with a Graphical User Interface (GUI) to simplify certificate management in a corporate context in a more intuitive manner.

3. Proposed PQC-Enhanced PKI Architecture

In this section, a strategy is proposed for integrating PQC algorithms into PKIs that are part of complex intra-and extra-company network ecosystems; showing an example of what it is considered as "complex" from an enterprise point of view and how integrating PQC algorithms can be difficult considering this complexity. A company is composed of as many applications as it offers services to its customers or employees; consequently, these applications need to communicate with each other to provide the service itself. The applications reside on a technological infrastructure composed of one or more servers, depending on their complexity, and therefore need to communicate with one another. Figure 1 represents, in a compact way, an application ecosystem and their possible communications. The ecosystem just described scales rapidly when considering external applications that reside outside the company and possess their own evolutionary logic from a cybersecurity perspective.

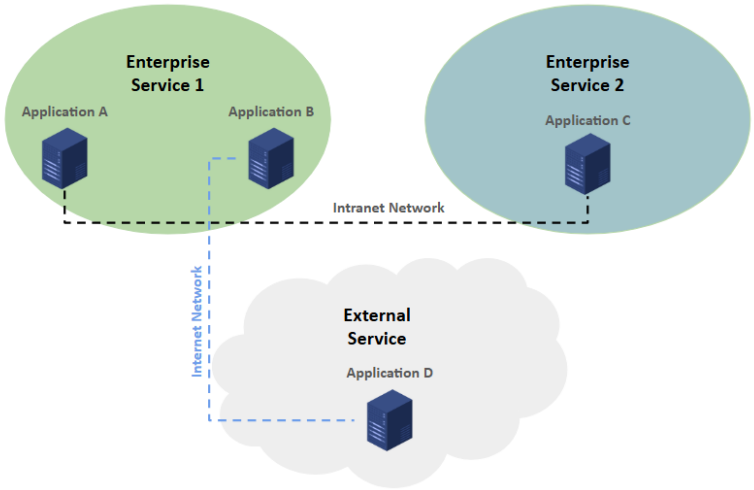

Figure 1: Applications composition of a service ecosystem and their intra-extra net communications.

For this reason, the adoption of hybrid certificates is highly recommended as the focal point of the crypto-agility approach. Hybrid certificates are a transitional security measure that supports both classical and post-quantum cryptography within a single digital certificate. They are designed to facilitate migration to quantum-safe systems, allowing for a gradual transition. That is, once the technological infrastructure underlying the applications is adequately updated to use PQC algorithms, it is possible to easily enable the use of these algorithms within communications, making them more secure.

In these terms, an enterprise can approach the securization of applications and communications in a targeted and isolated way to portions of infrastructure only when it has reached the maturity necessary to use PQC algorithms; vice versa, continue using common algorithms without cause interruption of services.

It is important to highlight the advantage of approaching the problem of technological adaptation in a modular and scalable way, in terms of the application portfolio managed by a company for which technological infrastructure synergy is necessary. Consequently, the same applies to infrastructure managed by third parties; migration is possible once the technological expertise is gained. The reason for the crypto-agility of these certificates is that they include both a traditional public key (such as RSA or ECC) and a post-quantum public key, each with its own signature.

To integrate a PKI into a complex corporate ecosystem, it is necessary to define and justify the placement of the PKI components within the corporate network. As illustrated in Figure 2, an abstract PKI architecture consists of three main components:

- Certification Authority (CA): an entity that is responsible for signing requests for the creation of digital certificates, formally known as Certificate Signing Requests (CSRs).
- Registration Authority (RA): a component for receiving requests and verifying their compliance based on the definition of internal or globally consolidated security standards.
- Validation Authority (VA): a service for verifying the validity of a digital certificate distributed by CAs.

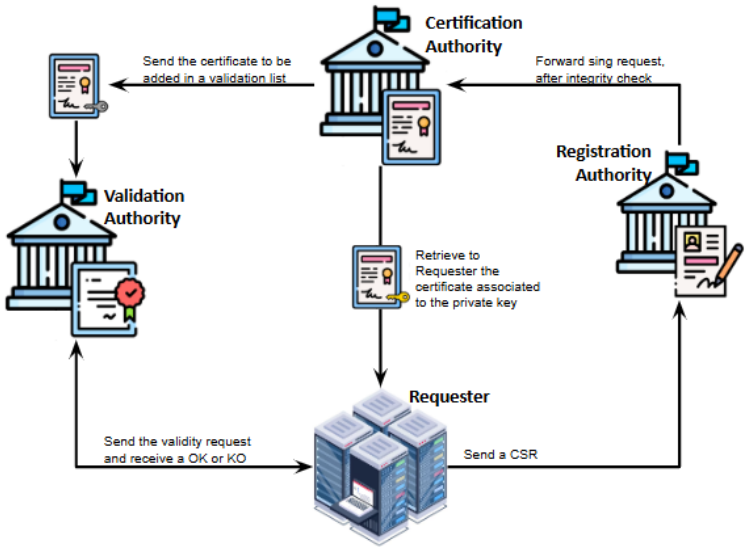

Figure 2: Abstract representation of a PKI architecture and relations between its components.

Having a single hierarchical level, in other words, a single entity for each PKI component, could result in a single point of failure for the organization should it become compromised. For this reason, according to security best practices, at least two hierarchical levels of Certification Authorities are typically used: a Root CA and an Intermediate (or Subordinate) CA. Since the latter is the only one exposed to potential attacks, its compromise would simply require replacement while keeping its root intact. Furthermore, by adopting a separation of duties, the second level can be replicated an arbitrary number of times according to internal needs; consequently, in this case, the transfer of the bottleneck to the Subordinate CA in the event of a compromise would also be avoided, as each second level CA is used for a specific purpose and replaced in the event of a compromise.

Figure 3 illustrates a hierarchy of CAs: a Root Certification Authority and several Intermediate CAs, each with a specific application purpose. All these Intermediate CAs are, in effect, formally the authority responsible for enforcing the signatures of digital certificate requests. This representation is the result

of experience gained over the years in the management of a large private corporate PKI, knowledge of the network architecture, the various separations of the corporate LAN, and the process of managing certificates issued by a private CA.

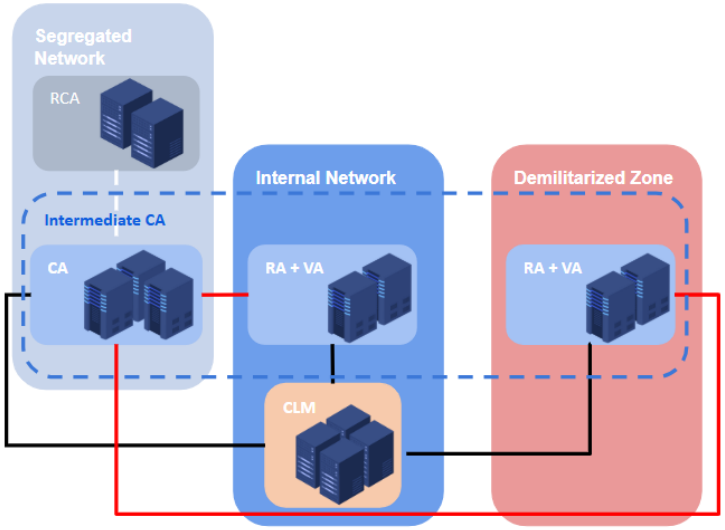

Figure 3: Architectural positioning of each element of a PKI architecture in a segmented network.

Also, a recommended usage in security best practices is that the Root CA must be located in a segregated network segment that is unreachable from any machine, as exfiltration could compromise communication reliability. A best practice for RCA may include that these servers must be offline and powered off. Intermediate CAs must be located in a dedicated portion of the internal network or in a shared portion with sensitive applications that must be protected in the same way; it is also important to pay attention to network configurations in order to communicate with these servers; only allowed and authorized parties should be able to interact with them. Finally, RA and VA services can be located on internal or exposed segments, depending on reachability requirements, and, if necessary, can be protected by third-party technologies, i.e., firewalls and proxies.

The CA component of a PKI can be configured to allow two digital signatures (a traditional one and a PQC-based one) to be used simultaneously. Using a chained digital signature approach would presuppose that the entire technological infrastructure is adequate and updated to accommodate the desired change; conversely, a hybrid approach allows for an efficient crypto-agile posture in terms of adapting the entire ecosystem surrounding the PKI.

4. Prototype Implementation

The practical implementation of the proposed architecture was carried out using two main open source software tools, each for its specific application area: the creation of hybrid digital certificates and communication via the TLS protocol.

The reason for this choice stems from the fact that the topic is still immature; thus, it has not been possible to identify a single solution that would allow both to be done centrally. Even with this approach, it is important to emphasize that the two tools aren't communicating natively, but they can be combined to achieve the desired goal.

The creation of a new Certification Authority was carried out using the open source EJBCA software [15] from KeyFactor. In their GitHub repository, the authors provide a free-to-use Dockerfile, which, although limited, is quite helpful. The reason for choosing this tool was twofold. On the one hand, its internal architecture is based on Kubernetes technology, making it much more versatile and scalable than traditional solutions, such as using the Windows AD CS server service. The other advantage is the ability to create hybrid certificates by combining standard signature algorithms with post-quantum

algorithms. In this case, the Dilithium algorithm was used. This is a much more agile solution than using nested, embedded encryption algorithms.

In the proposed implementation, a complete CA hierarchy was created, consisting of a Root CA and, below it, a Subordinate CA. The primary algorithm is RSA-4096, while Dilithium-44 is the secondary signature algorithm. The motivation for this choice is that, in the initial development phase, it is appropriate to maintain the classic signature algorithm; later, it will be possible to reverse the priority, retaining RSA as the fallback algorithm, with a technological infrastructure aligned to its use. Technically, this approach will add two additional OID extensions to the certificate, namely 2.5.29.73 and 2.5.29.74, which respectively indicate the alternative digital signature used and the alternative public key linked to the post-quantum algorithm.

As summarized in Figure 4, EJBCA enables the creation of a Certification Authority from scratch.

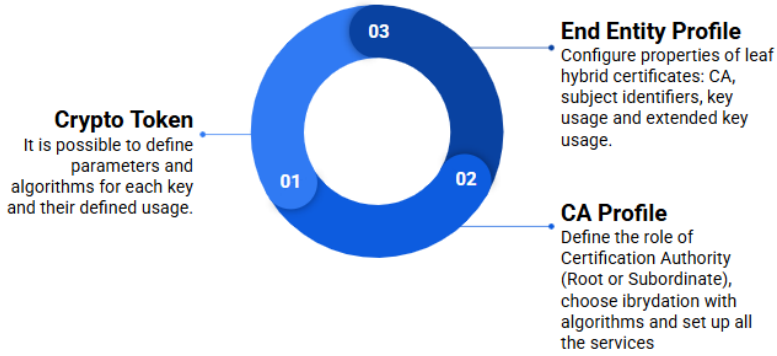

Figure 4: Logical process for CAs creation from scratch.

The creation workflow begins with defining the encryption tokens—the actual keys associated with the algorithms to be used. Furthermore, for each key, it is possible to define a purpose for which it should be used: encryption, signature, and testing; as shown in Table 1 with the selected algorithms and parameters. The latter key, in particular, is used to integrate an external component for the secure storage of these tokens (formally, a Hardware Security Module); this key is used to monitor the key's vitality status.

Table 1 comprehensively lists the keys and algorithms that are used and defined as application standards. This process was repeated for both the Root Certification Authority and the Intermediate Certification Authority, respectively. This rationale underpins a security standard by which the keys of the two CAs must be different to avoid dependent and functional ties in the event of a security breach.

Alias	Key Algorithm	Key Specification
encryptKey	ML-DSA	44
encryptKey-alt	RSA	4096
signKey	ML-DSA	44
signKey-alt	RSA	4096
testKey	RSA	2048

Table 1
Use cases of Crypto Token utilized, their algorithms and the parametrization choosed.

Next, one needs to create a Certification Authority profile that specifically defines the priority of the signature algorithms to be included in digital certificates. These algorithms must converge with the previously defined cryptographic tokens. In addition, further information can be defined that the Certification Authority must possess. The most important factors are the CA's validity (i.e., its End of Life), the distribution of the Certificate Revocation List (CRL), and the hierarchical path (i.e., how many CA levels are allowed below the one you created). This last setting is also important for strengthening security, as the goal is to prevent anyone from creating a subordinate CA signed by the compromised

one if the CA becomes compromised. Finally, it is possible to create the actual Certification Authority by combining the two objects defined in the previous steps.

Once a hierarchy has been configured, it is possible to define the final part of a PKI: the templates used to issue digital certificates, whether standard or hybrid, namely the End Entity Profile. In this case, it is possible to define which CA to associate with their issuance and which values must be defined during the creation phase to establish the digital identity, i.e., the Subject DN Identifiers. In this last aspect, it is possible to define or integrate hardening policies to limit which values can be modified and which must remain immutable.

An important component that could not be tested was the separation of the different CA services, i.e., creating separate containers for the Registration Authority and the Validation Authority. This feature can only be implemented using the Enterprise version of the software, which is available only to subscribed companies.

To integrate these algorithms into the TLS protocol, OpenSSL [16] was used, an open-source software that allows replication of client-server communication sockets while also integrating modules for the use of the algorithms standardized by NIST in its version 3.5.x.

The choice to use this software tool is due to the fact that, despite being a public project, it is well managed and maintained in the updated versions and bug patches of the released versions. In fact, many software applications integrate this software tool via the command line or within internal routines.

Recent versions of the software have introduced post-quantum algorithms, both in their pure form and in hybridizations with standard encapsulation algorithms such as RSA and elliptic curve cryptography. Furthermore, the power of this tool, and consequently the reason it was chosen, is the possibility of simulating a new communication channel by combining classic TLS with the Crystals-Kyber algorithm[3].

In particular, it is possible to define a client or server architecture by using the ”s_client” or ”s_server” configuration, respectively, within the openssl command line, followed by the ”-connect” option.

From the previous section on creating Certification Authorities, hybrid endpoint and CA digital certificates were used as objects for use within the simulation. Therefore, a dedicated repository, a key store, and a trust store were created, from which both certificates and private keys are retrieved via sockets. These can be accessed using the command line with the -key, -cert, and -CAfile options.

Finally, using the -groups parameter, it is possible to indicate the use of a specific algorithm for sharing symmetric keys. The Kyber algorithm was used by specifying MLKEM as the value, followed by its parameterization; in this case, 512 and 768 were used.

The commands are reported in the following, respectively, in an exemplary manner:

```
$openssl  s_server -cert  server.crt -key server.key -tls1_3 -accept 4433 -groups MLKEM512 -provider
 oqsprovider


$openssl s\_client -connect localhost:4433 -CAfile ca.crt -tls1\_3 -groups MLKEM512 -provider oqsprovider
```

Additionally, it is important to note that if you want to test the functionality of hybrid digital certificates, you can do so natively using OpenSSL alone, as it provides features that allow you to create Certification Authorities, generate asymmetric keys, and generate CSRs for digital certificates.

A major current limitation of OpenSSL is that, at the time of writing, it is virtually impossible to evaluate the crypto-agility of the two signature algorithms within digital certificates, because the functionality has not yet been developed and standardized as an IETF draft.

5. Performance Evaluation

This section presents the performance of the selected algorithms, both for digital signatures and for generating the symmetric keys used for message exchange. The metrics chosen to evaluate the algorithms are the time required for the handshake to complete and the number of bytes read and written during the same phase. These two metrics were chosen because the actual use of these algorithms could result in a burden on both the infrastructure, in terms of required memory, and the network, due to the need for greater byte volumes.

Indeed, the use of PQC signature algorithms increases the size of the keys used in communication. The incremental order is approximately 10 times higher, as shown in Table 5, compared to the RSA-2048 algorithm, which is currently used as a baseline for usage; although it could be higher depending on the parameterization of the Dilithium topology used.

To give an example of what is proposed, two PQC algorithms are considered: Crystals-Kyber[3] for key encapsulation and Crystals-Dilithium [4] for the digital signature. Specifically, for digital signature algorithms, the following algorithms are considered: RSA with a key length of 4096 bits and Dilithium with a parameter of 44. For key generation and exchange algorithms, both classical algorithms, particularly ECDH, and post-quantum algorithms, in this case MLKEM512 and MLKEM768, will be used. The objective of the evaluations is to measure the performance of two locally simulated sockets, expressed in terms of the time taken to complete the handshake phase, the recognition and validation of the certificate presented by the server, as well as the number of bytes read and written during the handshake phase.

Algorithm	Type	Public Key Size	Private Key Size	Signature Size
RSA-2048	Signature-Encryption	294 bytes	1190 bytes	256 bytes
RSA-4096	Signature-Encryption	550 bytes	2398 bytes	512 bytes
ECDSA-P256	Signature	64 bytes	32 bytes	64 bytes
Kyber-512	KEM	800 bytes	1632 bytes	-
Kyber-768	KEM	1184 bytes	2400 bytes	-
Kyber-1024	KEM	1568 bytes	3168 bytes	-
Dilithium-2	Signature	1312 bytes	2528 bytes	2420 bytes
Dilithium-3	Signature	1952 bytes	4000 bytes	3293 bytes
Dilithium-5	Signature	2592 bytes	4864 bytes	4595 bytes

Table 2
Exact key and signature sizes (in bytes) for the considered algorithms.

Below are examples of the two sockets, client and server, that were used to perform the performance evaluations by modifying the ”-groups” parameter in the OpenSSL command line for each iteration. To obtain more consistent tests, the SSL termination of the client with the server was performed 100 times, taking into account the average time recorded during each iteration. The script execution was saved in a bash file and assigned the necessary permissions with ”chmod”.

```
#SOCKET SERVER
openssl s_server -cert server_rsa.cer -key server_rsa.key \
    -tls1_3 -accept 4433   -groups MLKEM768

# SOCKET SERVER
#!/bin/bash

N=100
total_time=0
total_read=0
total_written=0

for i in $(seq 1 $N); do
    echo "Execution $i..."

    start=$(date +%s%3N)

    # Trace bytes in log file
    strace -f -s 512 -e trace=read,write,send,recv sh -c "echo | openssl s_client \
        -connect localhost:4433 \
        -CAfile SCARSA-chain.pem \
        -tls1_3 -groups MLKEM768 \
        -provider oqsprovider" \
        2> strace_run.log

    end=$(date +%s%3N)
    elapsed=$((end - start))
```

```
    # analysis trace log
    read_bytes=$(grep -E "read\(" strace_run.log | \
        awk '{ match($0, /read\([0-9]+, .*, ([0-9]+)\) = ([0-9]+)/, a); sum += a[2] } END { print sum }')

    written_bytes=0
    # sum write
    written_bytes=$(grep -E "write\(" strace_run.log | \
        awk '{ match($0, /write\([0-9]+, .*, ([0-9]+)\) = ([0-9]+)/, a); sum += a[2] } END { print sum }')
    # sum send
    send_sum=$(grep -E "send\(" strace_run.log | \
        awk '{ match($0, /send\([0-9]+, .*, ([0-9]+),/, a); sum += a[1] } END { print sum }')
    # sum recv
    recv_sum=$(grep -E "recv\(" strace_run.log | \
        awk '{ match($0, /recv\([0-9]+, .*, ([0-9]+),/, a); sum += a[1] } END { print sum }')

    if [ -n "$send_sum" ]; then
      written_bytes=$((written_bytes + send_sum))
    fi

    echo "Real Time: ${elapsed} ms, Bytes Read: ${read_bytes}, Bytes Write (write + send): ${written_bytes}"

    total_time=$((total_time + elapsed))
    total_read=$((total_read + (read_bytes + recv_sum)))
    total_written=$((total_written + written_bytes))
done

mean_time=$((total_time / N))
mean_read=$((total_read / N))
mean_written=$((total_written / N))

echo "--------------------------------"
echo "Mean real time: ${mean_time} ms"
echo "Mean byte read: ${mean_read}"
echo "Mean byte written: ${mean_written}"
```

Table 3 shows the performance results of the various signature algorithms combined with those for key exchange. As one might expect, using post-quantum algorithms while considering classical symmetric key generation algorithms requires more time, expressed in terms of milliseconds (ms). However, caution must be exercised when combining post-quantum algorithms, both for digital signatures and encapsulation, which are more efficient than the DS version using RSA but only slightly superior to the classical version currently available, with a time difference of less than 10 milliseconds.

	KEM Algorithm			
Signature Algorithm	Curve25519	P-256	MLKEM512	MLKEM768
RSA 4096	216	217	252	260
ML-DSA 44	226	222	224	226

Table 3
Time needed (expressed in milliseconds) for the SSL handshake to terminate.

Furthermore, in terms of memory used, as expected according to Table 2, which refers to the size of the keys, it emerged that the number of bytes read and written during the handshake phase is substantially increased when using PQC algorithms (Table 4).

	KEM Algorithm			
Signature Algorithm	Curve25519	P-256	MLKEM512	MLKEM768
RSA 4096	36298-7656	36336-7699	37034-8419	37359-8803
ML-DSA	44618-10591	44656-10634	45354-11354	45679-11738

Table 4
Amount of bytes read and written during the handshake process.

6. Conclusion

This article presents a practical proposal for the evolution and implementation of the current Public Key Infrastructure of a large company. It defines key points of focus, both in terms of the distribution of components within a corporate network and the adoption of post-quantum algorithms. The recommended solution is to use hybrid certificates that allow for two coexisting and independent signature algorithms, preferring the post-quantum one if the underlying application infrastructure is adequately updated to versions that support their use.

The topic, although rapidly evolving from an academic perspective, is still in its infancy when it comes to integrating it into a complex enterprise environment. However, we believe that what is discussed in this article can serve as an excellent starting point for the evolutionary process.

In the future, it would be interesting to further explore the topic of crypto-agility by practically defining what a framework could be for switching between PQC and classical algorithms, in line with PKI software evolution. Furthermore, in the medium to long term, it should be relevant to envision a possible corporate integration of QKD algorithms within the same framework. This would allow companies to further advance how symmetric communication keys are generated in a fully quantum-based manner, thus changing the paradigm of communications via the TLS protocol.

Regarding the proposed prototype, it would also be interesting, in the short term, to test the functionality of the portion reserved for the use of a paid version of the tool exclusively for companies. It would be possible to test the various technological and application integrations to evaluate the potential benefits from an automation perspective. The more complex an infrastructure, the greater the variety and number of certificates it will require, as well as their heterogeneity. Therefore, human effort is significant, both in terms of proper management and in terms of human error potential due to repetitive tasks. Adding a level of automation would, therefore, be crucial for successful integration.

Acknowledgments

This work was made possible thanks to the collaboration between Crédit Agricole Italia and the University of Parma. We thank Luigi Altavilla, Head of the Security Area; Giovanni Assolini, Head of IT Security Operations; Federico Giordani, Head of Human Resources at CAGS; and Ruggero Guidolin, General Manager.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

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

[2] NIST National Cybersecurity Center of Excellence, Migration to post-quantum cryptography, https://www.nccoe.nist.gov/crypto-agility-considerations-migrating-post-quantum-cryptographic-algorithms, 2025.

[3] NIST, Module-lattice-based key-encapsulation mechanism standard, https://-doi.org/10.6028/NIST.FIPS.203, 2024.

[4] NIST, Module-lattice-based digital signature standard, https://doi.org/10.6028/NIST.FIPS.204, 2024.

[5] NIST, Stateless hash-based digital signature standard, https://doi.org/10.6028/NIST.FIPS.205, 2024.

[6] D. Joseph, R. Misoczki, M. Manzano, J. Tricot, F. D. Pinuaga, O. Lacombe, S. Leichenauer, J. Hidary, P. Venables, R. Hansen, Transitioning organizations to post-quantum cryptography, Nature 605 (2022) 237–243.

[7] F. Bene, A. Kiss, Post-quantum security overview of the public key infrastructure, System Theory, Control and Computing Journal 3 (2023) 27–35.
[8] D. S. Alessio, T. Walter, C. Dajana, An adaptive dual-stack qkd-pqc framework for secure and reliable inter-site communication, Joint National Conference on Cybersecurity (ITASEC & SERICS 2025) 3 (2025) 27–35.
[9] IETF, RFC9881, https://www.rfc-editor.org/rfc/rfc9881, 2025.
[10] A. Wiesmaier, N. Alnahawi, T. Grasmeyer, J. Geißler, A. Zeier, P. Bauspieß, A. Heinemann, On PQC migration and crypto-agility, CoRR abs/2106.09599 (2021). URL: https://arxiv.org/abs/2106.09599. arXiv:2106.09599.
[11] M. Raavi, Q. Khan, S. Wuthier, P. Chandramouli, Y. Balytskyi, S.-Y. Chang, Security and performance analyses of post-quantum digital signature algorithms and their tls and pki integrations, Cryptography 9 (2025).
[12] J. Henrich, A. Heinemann, A. Wiesmaier, N. Schmitt, Performance impact of pqc kems on tls 1.3 under varying network characteristics, in: E. Athanasopoulos, B. Mennink (Eds.), Information Security, Springer Nature Switzerland, Cham, 2023, pp. 267–287.
[13] J. Henrich, N. Schmitt, N. Alnahawi, A. Heinemann, A lot of data and added complexity. how does pqc affect the performance of my tls connection?, in: S. K. Cha, J. Park (Eds.), Information Security, Springer Nature Switzerland, Cham, 2026, pp. 107–128.
[14] A. Mehmood, N. Tuveri, Integrating pqc in openssl via shallow providers for cryptographic agility, The 30th Nordic Conference on Secure IT Systems 30 (2025).
[15] KeyFactor, EJBCA, https://github.com/Keyfactor/ejbca-ce?tab=readme-ov-file, 2025.
[16] OPenSSL Software Foundation Inc., OpenSSL, https://github.com/openssl/openssl, 2025.