Ref. Ares(2024)5788163 - 12/08/2024

Horizon Europe

QUANTUM-ORIENTED UPDATE TO BROWSERS AND INFRASTRUCTURES FOR THE PQ TRANSITION (QUBIP)

# Analysis and design of PQ building blocks

**Deliverable number: D1.4**

Version 1.0

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

| | |
|---|---|
| **Project Acronym:** | QUBIP |
| **Project Full Title:** | Quantum-oriented Update to Browsers and Infrastructures for the PQ transition |
| **Call:** | HORIZON-CL3-2022-CS-01 |
| **Topic:** | HORIZON-CL3-2022-CS-01-03 |
| **Type of Action:** | HORIZON-IA |
| **Grant Number:** | 101119746 |
| **Project URL:** | https://www.qubip.eu |
| **Start date:** | 1 September 2023 |
| **Duration:** | 36 months |

| | | | |
|---|---|---|---|
| Editors: | Antonio Lioy | – | POLITO |
| Deliverable nature: | Report (R) | | |
| Dissemination level: | Public (PU) | | |
| Contractual Delivery Date: | 31 August 2024 | | |
| Actual Delivery Date | 12 August 2024 | | |
| Number of pages: | 80 | | |
| Keywords: | cybersecurity, post-quantum, cryptography | | |
| Contributors: | Andrea Vesco | – | LINKS |
| | Davide Margaria | – | LINKS |
| | Alberto Solavagione | – | LINKS |
| | Alessandro Pino | – | LINKS |
| | Grazia D'Onghia | – | POLITO |
| | Silvia Sisinni | – | POLITO |
| | Davide Bellizia | – | TELSY |
| | Agostino Sette | – | TELSY |
| | Alberto Battistello | – | SECPAT |
| | Maria Chiara Molteni | – | SECPAT |
| | Eros Camacho-Ruiz | – | CSIC |
| | Piedad Brox | – | CSIC |
| | Antonio Pastor | – | TID |
| | Diego Lopez | – | TID |
| | Ulises Pastor | – | TID |
| | Javier Faba | – | UPM |
| | Juan Pedro Brito | – | UPM |
| | Nicola Tuveri | – | TAU |
| | Daniel Luoma | – | TAU |
| | Akif Mehmood | – | TAU |
| | Alex Shaindlin | – | TAU |
| | Dmitry Belyavskiy | – | REDHAT |
| | Sahana Prasad | – | REDHAT |
| Peer review: | Andrea Pozzi | – | SMART |
| | Javier Faba | – | UPM |
| Security review: | Estanislao Fernández | – | TID |
| | Juha Nurmi | – | TAU |
| Approved by: | ALL partners | | |

**Table 1:** Document revision history

| Issue Date | Version | Comments |
|---|---|---|
| 21/03/2024 | 0.1 | Initial table of contents |
| 10/06/2024 | 0.2 | First draft version, for internal review |
| 20/06/2024 | 0.9 | Second draft version, for review by SAB |
| 12/08/2024 | 1.0 | Final version, for submission |

## Abstract

This document represents the Deliverable D1.4 of the Quantum-oriented Update to Browsers and Infrastructures for the Post-quantum transition (QUBIP) project. It describes the requirements and the design of the Post-Quantum (PQ) building blocks. They will be integrated into the three different pilot demonstrators.

# Contents

# List of Figures

# List of Tables

# List of Acronyms

| | |
|---|---|
| **ABI** | Application Binary Interface |
| **AES** | Advance Encryption Standard |
| **ANSSI** | Agence Nationale de la Sécurité des Systèmes d'Information [French Cybersecurity Agency] |
| **APB** | Advanced Peripheral Bus |
| **API** | Application Programming Interface |
| **ATF** | ARM Trusted Firmware |
| **AXI** | Advanced eXtensible Interface |
| **BLNS** | Bootle, Lyubashevsky, Nguyen, and Sorniotti |
| **BOM** | Bill-Of-Materials |
| **BSI** | Bundesamt für Sicherheit in der Informationstechnik [Federal Office for Information Security] |
| **CA** | Certification Authority |
| **CCN** | Centro Criptológico Nacional |
| **CCIPS** | Centrally Controlled IPSec |
| **CLI** | Command Line Interface |
| **CNSA 2.0** | Commercial National Security Algorithm Suite 2.0 |
| **CPU** | Central Processing Unit |
| **CRL** | Certificate Revocation List |
| **CRQC** | Cryptographically Relevant Quantum Computer |
| **CRTM** | Core Root of Trust for Measurement |
| **CSR** | Certificate Signing Request |
| **CTS** | Commit-Transferable Signatures |
| **DID** | Decentralized IDentifier |
| **DLT** | Distributed Ledger Technology |
| **DTLS** | Datagram Transport Layer Security |
| **ECC** | Elliptic-Curve Cryptography |
| **EMV** | Europay, Mastercard, and Visa |
| **EU** | European Union |
| **FFI** | Foreign Function Interface |
| **FPGA** | Field Programmable Gate Array |
| **FSBL** | First Stage Bootloader |
| **fTPM** | firmware Trusted Platform Module |
| **GSMA** | Global System for Mobile communications Association |
| **GUI** | Graphical User Interface |
| **HBS** | Hash-Based Signature |
| **HSM** | Hardware Security Module |
| **HW** | Hardware |
| **I2C** | Inter Integrated Circuit |
| **I2NSF** | Interface to Network Security Function |
| **IETF** | Internet Engineering Task Force |
| **IKE** | Internet Key Exchange |
| **IMA** | Integrity Measurement Architecture |
| **IoT** | Internet of Things |
| **IP** | Internet Protocol |
| **IPsec** | IP Security |

| | |
|---|---|
| $\mathsf{ISIS}_f$ | Inhomogeneous $\mathsf{SIS}_f$ |
| **JWK** | JSON Web Key |
| **JSON** | JavaScript Object Notation |
| **KDF** | Key Derivation Function |
| **KEM** | Key Encapsulation Method |
| **LDPC** | Low Density Parity Check |
| **LMS** | Leighton-Micali Hash-Based Signature |
| LWE | Learning with Errors |
| **MCU** | Micro-Controller Unit |
| **MiTM** | Man-in-The-Middle |
| **ML-DSA** | Module-Lattice-Based Digital Signature Algorithm |
| **ML-KEM** | Module-Lattice-Based Key-Encapsulation Mechanism |
| MLWE | Module Learning with Errors |
| **MPU** | Micro-Processor Unit |
| MSIS | Module Short Integer Solution |
| **NIST** | National Institute of Standards and Technology |
| **NIZK** | Non-Interactive Zero-Knowledge proof |
| **NSA** | National Security Agency |
| **NSF** | Network Security Function |
| **NSPR** | Netscape Portable Runtime |
| **NSS** | Network Security Services |
| **NTRU** | N-th degree Truncated polynomial Ring Units |
| **OCSP** | On-line Certificate Status Protocol |
| **OP-TEE** | Open Portable Trusted Execution Environment |
| **OQS** | Open Quantum Safe |
| **OS** | Operating System |
| **PCB** | Printed Circuit Board |
| **PCR** | Platform Configuration Register |
| **PKC** | Public-Key Certificate |
| **PKCS** | Public-Key Cryptography Standards |
| **PKI** | Public-Key Infrastructure |
| **PL** | Programmable Logic |
| **PoKS** | Proof of Knowledge of a Signature |
| **PQ/T** | Post-Quantum/Traditional |
| **PQ** | Post-Quantum |
| **PQC** | Post-Quantum Cryptography |
| **PQTN** | Post-Quantum Telco Network |
| **PRNG** | Pseudo Random Number Generator |
| **PS** | Processing System |
| **QKD** | Quantum Key Distribution |
| **QoS** | Quality of Service |
| **QUBIP** | Quantum-oriented Update to Browsers and Infrastructures for the Post-quantum transition |
| **RA** | Remote Attestation |
| **RAM** | Random Access Memory |
| **RCA** | Root Certification Authority |
| **ROM** | Read Only Memory |
| **RoT** | Root of Trust |
| **RSA** | Rivest–Shamir–Adleman |
| **S/MIME** | Secure/Multipurpose Internet Mail Extensions |
| **SA** | Security Association |

| | |
|---|---|
| **SAD** | Security Association Database |
| **SCP** | Secure Channel Protocol |
| **SDN** | Software-Defined Networking |
| **SE** | Secure Element |
| **SIM** | Subscriber Identity Module |
| SIS | Short Integer Solution |
| **SLH-DSA** | Stateless Hash-Based Digital Signature Algorithm |
| **SoC** | System-on-Chip |
| **SotA** | State of the Art |
| **SPD** | Security Policy Database |
| **SSI** | Self-Sovereign Identity |
| **SW** | Software |
| **TC** | Trusted Computing |
| **TCG** | Trusted Computing Group |
| **TEE** | Trusted Execution Environment |
| **TF-A** | Trusted Firmware-A |
| **TF-M** | Trusted Firmware-M |
| **TLS** | Transport Layer Security |
| **ToT** | Triangle-of-Trust |
| **TPM** | Trusted Platform Module |
| **TTP** | Trusted Third Party |
| **URI** | Uniform Resource Identifier |
| **VC** | Verifiable Credential |
| **VDR** | Verifiable Data Registry |
| **VP** | Verifiable Presentation |
| **VPN** | Virtual Private Network |
| **W3C** | World Wide Web Consortium |
| **WNS** | Weak Non-Separability |
| **WP** | Work Package |
| **XMSS** | eXtended Merkle Signature Scheme |
| **ZK** | Zero-Knowledge |

# 1. Introduction

The deliverable D1.4 “Analysis and design of PQ building blocks” is issued at M12 (i.e., August 2024). Its scope is to describe the analysis performed and the suggested design for the building blocks of the project, that are those later used in three pilot demonstrators.

In each section, a specific building block (often intended as a “class”) is introduced. Then it is analyzed with respect to critical issues for PQ transition, so that proper requirements can be derived. This may also involve considering current scientific, technical, and commercial trends for this kind of operation. Finally, a possible design (that might be not the unique one) to satisfy these requirements is described.

Considering the EU’s commitment to stringent data protection and robust cybersecurity, the NIST selection of Post-Quantum Cryptography (PQC) standards is just one part of the picture. European and national agencies also offer their own sets of recommendations for the PQC transition, reflecting a diverse range of strategies and priorities across Europe. This variety ensures that cryptographic security approaches are tailored to different regional needs and perspectives on emerging quantum threats.

In this context, *cryptographic agility* is increasingly critical. Cryptographic agility refers to the ability of a product or system to support updating its cryptographic algorithms without recalling it or replacing it with a new one. Moreover, *pliability* represents another relevant characteristic of the transition. Pliability refers to the need for the transition to adapt to network management best practices, support established network services, and be integrated through standardized approaches.

These capabilities are essential to ensure that digital infrastructures remain secure and resilient against both current and future threats, and even more relevant considering systems which may be deployed under different regional recommendations. For example, Spain’s CCN has issued its own recommendations [4]. Similarly, Germany’s BSI and France’s ANSSI each regularly update their respective recommendations on the PQC transition [5, 6, 7]. Both agencies are currently mostly aligned in their latest positions, but exhibit some differences from the algorithm selection operated by NIST [8] and NSA [9], in the portfolio of algorithms (e.g., FrodoKEM [10] and Classic McEliece [11] are featured in their Key Encapsulation Method lists, but not in NIST’s current selection), in the parameter sets (e.g., the BSI indicated they might include only parameter sets corresponding to NIST security levels 3 and 5 [7]), or about Post-Quantum/Traditional (PQ/T) Hybrid considerations. For these reasons, agencies in Europe currently promote cryptographic agility, in order to better protect digital sovereignty and the security of European citizens’ data.

Throughout this document, the keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL", are to be interpreted as described in RFC-2119 [12].

This project has received funding from the European Union under the
Horizon Europe framework programme [grant agreement no. 101119746].

# 2. Public-Key Infrastructure

## 2.1. Introduction

Public-Key Infrastructure (PKI) is at the heart of several modern security solutions, from digital signature for electronic documents to peer authentication in secure network channels – such as IP Security (IPsec) or Transport Layer Security (TLS) – from authentication of software updates to identity of devices. PKI is a technical and management infrastructure for the creation, distribution, and validation of Public-Key Certificates (PKCs). A PKC is a data structure that securely binds a public key with the identity and/or attributes of the owner of the corresponding private key. The transition exercise is represented by the integration of PQ digital signature algorithms in the current PKI and the evaluation of their impact in terms of both performance and storage.

This chapter starts with an overview of the main components of PKI, namely Certification Authority (CA), Certificate Revocation List (CRL) and On-line Certificate Status Protocol (OCSP) protocol. Afterward, the requirements for PQ PKI are provided. However, in order to perform the PQ transition, it is important to mention and follow the recommendations from official organizations. Finally, a design choice for PQ certificates is presented.

### CA

The CA is the entity which builds the trust basis for PKI by issuing and revoking PKCs. When implementing a PKI, an organization can either deploy its own CA or rely on a Trusted Third Party (TTP).

A valid chain of trust is composed by three entity classes: (1) the Root Certification Authority (RCA), whose certificate is self-signed with the RCA's private key; (2) the Intermediate CAs, that extend trust to End-entities; (3) End-entities, that provide critical information to the issuing CA via a Certificate Signing Request (CSR) form. The certificate is then signed and issued by a trusted CA.

Certificate validation plays an essential role in PKI and it requires not only cryptographic validation of the PKC itself and its trust chain, but also verification that each certificate in the chain was not revoked at the time it was used to sign a certificate or a document.

### CRL

A CRL is a signed data structure that contains a list of revoked certificates. A CRL can be requested once to the corresponding CA and cached locally to be read without having access to Internet. The publication of a CRL is done periodically, thus allowing to obtain the full revocation-related information at specific points in time, even when the relying party is offline. However, this periodic public mechanism may have some disadvantages:

- CRLs could become big and consulting them would be potentially time-consuming;
- a lot of storage might be needed to store CRLs, which is a problem in low-memory applications such as mobile or Internet of Things (IoT) devices;
- the validity period and publications of CRL is in the order of days, and the information available may be not up-to-date.

For this reason, OCSP protocol is used.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

### OCSP

The OCSP protocol allows clients to query an OCSP server about the revocation status of individual certificates, providing more up-to-date information with respect to a CRL. OCSP does not require much storage, but requires applications to be online. The OCSP server responds with a digitally signed message containing the actual status of the certificate. In order to achieve more end-user privacy in Internet browsing, OCSP-stapling can be included during the verification of the status of certificates.

## 2.2. Requirements

The transition of PKI to PQC involves specific requirements focused on accommodating PQ algorithms. Standardization of these algorithms must align with their integration into existing protocols. Firstly, the security infrastructure must accommodate PQ algorithms. This involves ensuring that the PKI can handle the larger key sizes and signatures associated with these algorithms. RSA-2048 has a public key and signature size of $256\,\mathrm{B}$, while ECC algorithm with the same security level has key size and signature size of respectively $28\,\mathrm{B}$ and $64\,\mathrm{B}$. PQ algorithms have much different parameters. For example, FALCON 1024 [13] has a public key size of $1793\,\mathrm{B}$ and a signature size of $1280\,\mathrm{B}$. These parameters significantly exceed those of traditional RSA or ECC keys, necessitating adjustments in storage, processing power, and transmission efficiency. Secondly, the implementation should maintain compliance with performance and speed requirements, especially in certificate generation and chain verification processes. These performance metrics are critical for maintaining operational efficiency and user experience.

Additionally, as PQ algorithms may have different operational characteristics, it is essential to review and update security policies and practices to address new potential vulnerabilities and attack vectors introduced by the shift to PQC.

Finally, compliance with official guidelines is critical in the design and implementation of PQ PKI systems.

The main requirements for a PQ PKI are summarized in Table 2.1.

**Table 2.1:** PQ PKI Requirements

| Req. | Description |
|---|---|
| PKI-01 | The Root CA, intermediate CAs and end-entities MUST ensure cryptographic agility |
| PKI-02 | `Public Key Algorithm` field of PQ certificate MUST support PQ algorithms identifiers |
| PKI-03 | `Public Key` field of PQ certificate MUST NOT have size limits |
| PKI-04 | Issuer Key MUST be strong and MUST NOT have size limits |
| PKI-05 | Signature algorithm's speed MUST be considered if certificates are generated on the fly |
| PKI-06 | Signature applied by the CA on the certificate MUST be strong and short |
| PKI-07 | CRL's Issuer Key MUST be strong |
| PKI-08 | Signature verification speed SHOULD be comparable to classical signature algorithms |

It is important to outline a scale of priorities for both performance and storage requirements, since PQ algorithms will inevitably introduce challenges from this point of view. Verification speed is more important in end-entities (that sometimes has constrained resources), whilst signature generation speed is more

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

important for servers that sign certificates. In general, the following order SHOULD be considered to choose PQ algorithms for PKI, with performance at least comparable with the classical algorithms:

1. Signature verification speed, which is crucial for end-entities that sometimes lack of powerful resources and are in charge of verify the certificates. This aspect is also significant for OCSP protocol;
2. Signature generation speed, which is important for servers;
3. Key generation speed, which can be improved through Hardware Security Module (HSM).

On the other hand, the sizes of public keys, private keys and signatures MUST be considered because of their influence on the total size of the certificate, with the following priorities:

1. Certificate size;
2. Public key size, which is crucial for Intermediate CAs since they are in charge of transferring certificate chains. Meanwhile, Root CAs can handle bigger public keys;
3. Private key size, which is important for systems with limited storage.

## 2.3. Current Trends

This section aims to outline the starting point for a design choice, based on official guidelines, recommendations and existing implementations.

### 2.3.1. Recommendations

#### IETF

The IETF suggests that quantum-safe authentication is achieved through either a pure PQ or a PQ/T hybrid Certificate [14]. A pure PQ X.509 Certificate should use Module-Lattice-Based Digital Signature Algorithm (ML-DSA) with the most appropriate version based on the needed security level. In order to decide which certificate to adopt, it is important to consider some factors. For example, for applications with short key lifetimes it might be acceptable to still consider only classical algorithms, because a quantum attack would take longer to run than the lifespan.

The IETF recommends that the first step could be to have a heterogeneous PKI where only long-lived CAs are using PQ algorithms.

#### CA/Browser Forum

Even if the CA/Browser Forum did not provide yet recommendations about the PQ transition of PKI, it is still worth mentioning the last ballot about reducing TLS server certificate lifetimes [15]. The consequences of this change (which is going into enforcement during 2024) might influence the design decision for PQ transition as well. These changes aim to manage the increasing number of machine identities, especially in containerized and cloud-native environments.

The strong distinction between short-lived and long-lived certificates makes them subject to different requirements. The lower lifetime is going to impact especially short-term certificates, while long-term will not be much affected. Since short-lived certificates will have a lifecycle of 7 days in 2026, they will not be associated to any revocation information. Therefore, the CA will no longer be in charge of revoking this kind of certificates, and OCSP will become optional. On the other side, long-lived certificates will still be associated to CRLs.

As a consequence, shorter certificate lifespans increase the frequency of certificate issuance and renewal processes, necessitating efficient and scalable PQC algorithms that can handle larger key sizes and signatures without compromising performance. Organizations will need to adopt automation tools and robust

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

infrastructure to manage the increased volume of certificates and ensure smooth integration of PQC into their existing PKI systems.

#### BSI

The German BSI suggests that PQ algorithms standardization should happen in parallel with the integration of the new algorithms in the existing cryptographic protocols [16]. Moreover, PQ algorithms should be used only in combination with classical ones, at least in the first phase of the transition, in order to ensure compatibility.

About PKI, the BSI suggests to start with hybrid X.509v3 certificates with PQ keys and signatures marked as non-critical extensions. However, if an application supports or is aware of the new extensions, only the PQ part of the certificate should be verified, otherwise there would not be an actual improvement in security. Finally, according to BSI, hash-based signature schemes, such as Leighton-Micali Hash-Based Signature (LMS) [17] and eXtended Merkle Signature Scheme (XMSS) [18], are more suitable for long-lived root certificates, as explained in [19].

Hash-based signature schemes have reduced key and signature size but are stateful, which means that private keys must always be handled in a secure environment. Root CAs can provide such a secure environment and the number of issued signatures is not high because many root CAs delegate OCSP services or the signing of end-entity certificates to external entities that implement stateless signature schemes. In this way, the security issues that might arise from the usage of stateless algorithms are mitigated.

#### NSA

In the Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) [20], different PQ algorithms are chosen for different use cases:

1. **Software and Firmware updates** requires LMS [17] and XMSS [21] algorithms;
2. **Symmetric encryption** requires either SHA-384 or SHA-512 for integrity and AES-256 as encryption algorithm;
3. **Asymmetric encryption** should follow the standardization process conducted by NIST. The recommended algorithms from the NSA are ML-DSA ad Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM).

#### GSMA

The GSMA Post-Quantum Telco Network (PQTN) Task Force has provided comprehensive best practice guidelines for transitioning to quantum-safe solutions [22]. Although there is no focus on PKI, these guidelines include important implications for its deployment in telecommunications. The most relevant recommendations for PQ PKI regard automation and standardization.

Automation is critical for adopting cryptographic agility and quantum-safe solutions at scale. It supports various stages of the PQ transition, ensuring efficiency and consistency. Additionally, utilizing standardized algorithms, protocols, and solutions facilitates migration and minimizes costs. Standardization ensures broad compatibility and eases the integration of new cryptographic methods.

### 2.3.2. Existing Implementations

It is important to evaluate existing implementations in order to understand the current status of the technology. Moreover, existing implementation may be considered as starting point for further contributions.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

Several open-source implementations showcase the integration of PQ algorithms into PKI:

- **GlobalSign Repository** [23] contains examples of PQ certificates, CRLs, and OCSP requests and responses.
- **Open Quantum Safe (OQS) Project** [24] allows the generation of PQ X.509 certificates using a modified version of OpenSSL 1.1.1 or standard OpenSSL 3 with the OQS provider. The project supports generating these certificates with the common OpenSSL 'x509' commands.
- **IETF Hackathon** [25] provides a repository of X.509 data structures using PQ and composite algorithms, combining classical and PQ cryptography.

These initiatives demonstrate practical steps towards integrating PQ algorithms into existing PKI frameworks, ensuring long-term security against quantum computing threats.

## 2.4. Design

There are some external factors that create constraints in the design of the solution, therefore should be considered. These factors can be summarized in:

1. **Hardware readiness**, namely the status of current hardware systems with respect to PQ transition;
2. **Software readiness**, namely being aware of the existing software that are currently taking steps forward in terms of transition to PQC. Examples are Chrome 124 [26] and CloudFlare [27];
3. **Certificate readiness** in terms of full control of the certificates, which is an essential requirement for a complete transition to PQC. This aspect should be considered in parallel with software readiness.

### Certificate Agility

In general, the transition to PQ PKI should follow the principle of cryptographic agility [22, 28], shaped in the form of certificate agility. The latter is the ability to replace certificates without errors and in the smoothest way. One way to achieve certificate agility is to adopt automation solutions for automatic provision, deployment, and expiration actions.

### Hybrid Certificates

Design considerations necessitate intermediate steps before transitioning to pure PQ certificates [29]. The initial phase involves deploying hybrid certificates [30], in order to have backward compatibility. These certificates incorporate both classical and PQ cryptographic elements, with PQ keys, algorithms, and signatures integrated into the `Subject Alt Public Key Info`, `Alt Signature Algorithm`, and `Alt Signature Value` fields as non-critical extensions of the X.509v3 format, while classical algorithms remain in the `tbsCertificate` element. In this way, the same certificate provides two or more signatures and keys, which is a default feature of the X.509v3 format. It is important to note that, if an application has PQC support to verify a hybrid certificate, only the PQ signature should be considered, otherwise there would not be PQ transition at all. Hybrid cryptographic schemes can provide transitional and fallback mechanisms. However, they may introduce computational and complexity overheads that could be unsuitable for some contexts [22]. Therefore, when certificate size is a serious issue, the PQ part of the certificate can be embedded in `DeltaCertificateDescriptor` extension, which allows to encode the differences between two parallel certificates [31].

For security-critical applications, such as CA, the next step involves using composite certificates. The latter use ASN.1 encoding to separate the traditional and PQ objects [32]. Ultimately, pure PQ certificates can be implemented once the systems are fully upgraded to PQ algorithms [29]. Currently, conventions for using the ML-DSA in Internet X.509 certificates and CRLs are described in [33].

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

### Algorithms selection

Since in the QUBIP project PKI for Internet Browsing and IoT-based Digital Manufacturing use cases are taken into account, different certificates are going to be implemented. Internet Browsing requires X.509 "core" hybrid certificates with public CA, while private CAs and "specialized" certificates can be implemented for IoT authentication. "Core" certificates must follow the migration paths provided by the platform providers. On the other hand, "specialized" certificates are not supposed to follow a precise path, thus being an easier PQ migration context. The PQ digital signature algorithms will be selected according to the requirements expressed in Section 2.2:

- Root CAs MUST adopt an algorithm with at least security level 3, according to the NIST's evaluation criteria for security levels [34]. Based on the scales of priorities and the experimental evaluation in Table B.1 in Appendix B, the most suitable choice is ML-DSA. Long-lived root certificates may also adopt Stateless Hash-Based Digital Signature Algorithm (SLH-DSA);
- ML-DSA with security level 2 is suitable for intermediate CAs and end-entities.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

# 3. Integrity Verification

## 3.1. Introduction

In today's digital landscape, ensuring the integrity and authenticity of computing systems is paramount, so Integrity Verification techniques are crucial features of a secure computing environment. Integrity Verification encompasses mechanisms such as secure boot, which prevents unauthorized code from executing during startup, measured boot, which records the integrity state of each component before its execution during the boot process, and runtime measurement, which ensures the system remains unaltered during its operation.

Remote Attestation, on the other hand, allows a system to provide verifiable proof of its integrity status to a remote entity, facilitating trust in distributed and cloud-based environments. Together, these mechanisms form a robust defense against unauthorized modifications and intrusions. The necessity for these security measures stems from the increasing sophistication of cyber threats. Attackers continuously devise new methods to compromise systems by altering boot sequences, injecting malicious code during runtime, or altering system configurations.

The advent of Cryptographically Relevant Quantum Computer (CRQC) poses a significant challenge to validity of Integrity Verification mechanisms, which widely use cryptographic algorithms such as RSA and ECC, threatened by the computational capabilities of quantum computers. As quantum technology advances, the urgency to transition to quantum-resistant cryptographic procedures becomes evident, especially with regard to the mechanisms that represent the basis of trust in modern digital infrastructures.

Migrating Integrity Verification techniques to quantum-safe approaches involves selecting cryptographic algorithms that can withstand the capabilities of CRQCs, and integrating them in secure boot, measured boot, runtime integrity, and remote attestation processes. This demands a comprehensive overhaul of key management practices, cryptographic libraries, and protocol standards used into existing Integrity Verification frameworks. The transition of Trusted Computing procedures against the imminent revolution of quantum computing is a complex exercise that must be addressed urgently, since it is considered a priority by the scientific community compared to other security techniques.

Regarding Integrity Verification, the QUBIP project aims to analyze the security requirements, performance implications, and technical constraints related to deploying PQC solutions in these critical security services. In the following, sections 3.2 and 3.3 will address the analysis of system requirements and design building blocks needed to carry out transition of Integrity Verification techniques to PQC on different device categories: embedded systems, in the IoT-base Digital Manufacturing pilot demonstrator, and network platforms, in the Software Network Environments for Telco Operators pilot demonstrator.

## 3.2. Requirements

PQ Integrity Verification needs specific technical, performance, and implementation requirements tailored to different target platforms. Table 3.1 lists general requirements needed to enable PQ Integrity Verification techniques across platforms.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

**Table 3.1:** PQ Integrity Verification Requirements

| Req. | Description |
|---|---|
| IV-01 | PQ Integrity Verification MUST rely on PQ cryptographic primitives |
| IV-02 | Each component involved in the boot sequence of the system MUST be signed by a trusted authority with quantum-resistant digital signature algorithms |
| IV-03 | PQ digital signature algorithms adopted for PQ Secure Boot MUST support rapid signature verification; factors like signature size, key size, and computational complexity of the signature verification operation MUST be considered |
| IV-04 | PQ digital signature algorithms adopted for implementing remote attestation MUST support rapid signature generation to ensure the shortest attestation cycles possible |
| IV-05 | PQ digital signature algorithms adopted for implementing remote attestation SHOULD support rapid key generation and signature verification. |
| IV-06 | Platform components, encompassing firmware, operating system, and applications, MUST adopt post-quantum secure hash functions for acquiring integrity measurements on binaries and configuration settings that impact security |
| IV-07 | PQ Integrity Verification MAY adopt hybrid cryptographic approaches that combine classical and post-quantum algorithms, to allow seamless integration into existing systems and protocols |
| IV-08 | PQ Integrity Verification procedures SHOULD support crypto-agility, in order to allow for easy updates as new PQ algorithms become available |
| IV-09 | PQ Integrity Verification implementation SHALL comply with specifications and standards set by recognized standards bodies such as NIST, IETF, TCG, and with security guidelines relevant to the target platform and deployment environment |

## 3.3. Design

This section outlines the design of the components of the Integrity Verification building block necessary for migrating Integrity Verification protocols to PQC. The aim is to ensure the long-term security of these procedures against the potential threats introduced by quantum computing. In the following, high-level design objectives are presented.

### 3.3.1. Secure Boot

Secure boot ensures that a device boots using only software that is trusted by the device manufacturer and/or device owner.

Making the secure boot process quantum-safe requires first integrating PQC at the firmware level, in order to ensure that the initial code executed during the boot process is authenticated using PQ signature algorithms. There are some key factors that need to be considered when choosing the optimal digital signature algorithm to use for PQ secure boot.

First, PQ digital signatures used for secure boot require a high *security strength* and *algorithm maturity*, as they should remain valid for long periods of time. The security of Hash-Based Signature (HBS) schemes relies solely on the properties of the underlying hash functions, which are well-studied and understood in cryptography, providing a solid foundation for their security compared to newer approaches, such as

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

lattice-based ones [35]. This consideration leads to preferring HBS schemes in the implementation of PQ secure boot. Moreover, when choosing PQ signature algorithms, it would be useful to prefer those that are already part of the NIST PQC standardization process, or have been recommended by other reputable standardization bodies (e.g., IETF). Among HBS schemes, NIST selected SLH-DSA for standardization, while the IETF developed RFCs for two Stateful HBS schemes, LMS [17] and XMSS [21]. NIST has also published a recommendation document [36] on the use of Stateful HBS schemes.

Secure boot also requires a careful evaluation of *resource requirements* and *performance*, as the PQ signature algorithms have to be efficient in the signature verification operation in order to keep boot time and system latency low. Regarding resource-constrained devices, like embedded systems used in the IoT-based Digital Manufacturing pilot demonstrator, a careful evaluation of the memory consumption is necessary in terms of the size of keys, signatures and any auxiliary data structures, as these devices typically have limited memory and storage.

Furthermore, it is crucial to ensure that the integration of PQ algorithms does not introduce new vulnerabilities in the secure boot process. Stateful HBS algorithms (LMS, XMSS) require secure and reliable mechanisms for state storage and updates to ensure their security and correct operation, therefore their adoption would add complexity. However, when these algorithms are used to implement secure boot, they do not increase the implementation complexity in the device, as the management and updating of the state have to be carried out by the entity that signs the software, external to the device. In contrast, Stateless HBS schemes (SLH-DSA) allow avoiding state management, but require efficient handling of larger signatures, and often have slower verification performance as they require more complex operations.

Based on the previous considerations, Stateful HBS algorithms, like LMS and XMSS, represent an optimal choice for secure boot use case, particularly for resource-constrained devices. XMSS shares similarities with LMS and comes with a tighter security proof than LMS. Several studies have demonstrated its applicability in constrained devices [37, 38, 39]; however, based on [3, 40], it has worse performance than LMS. For this reason, LMS is the algorithm selected as the first choice for implementing secure boot, as recommended also by the CNSA 2.0 [20]. A performance comparison of HBS algorithms can be found in Appendix D. On the other side, for measured boot and remote attestation, SLH-DSA is a better choice due to its stateless nature.

### 3.3.2. Measured Boot

Measured boot allows to create and maintain a chain of trust from the moment the system is powered on until the operating system is fully loaded. During the boot flow of the platform, each critical system component (e.g., firmware, bootloaders, operating system kernel) is measured by the previous one, before it gains control of the platform; the measurement is typically a cryptographic digest computed with a secure hash function.

The measured boot chain of trust starts with a Core Root of Trust for Measurement (CRTM), which is typically implemented with a secure, immutable component, contained in Read Only Memory (ROM). Due to its immutable nature, the transition exercise will not affect the CRTM of the target devices.

Implementing a PQ measured boot requires adopting PQ secure hash functions to generate cryptographic hashes of code and configurations. While quantum computers especially threaten classical asymmetric cryptographic algorithms, their impact on hash functions is less severe, but still significant. The security strength of a hash function against quantum attacks is primarily evaluated based on Grover's algorithm [41], which can speed up the brute-force search for pre-image attacks, effectively reducing the security level of the hash function by half [42]. For collision resistance, quantum computers do not provide a significant speedup over classical attacks; the best-known quantum attack does not significantly improve over the classical birthday attack, which halves the security strength [43]. Thus, if SHA-256 is currently considered to have acceptable security strength against classical attacks, SHA-512 or SHA-3-512 should

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

be adopted in PQ measured boot, in order not to lower the security strength of this process due to Grover's algorithm.

The measurements acquired during the boot stages have to be recorded in a secure storage, which usually is the Trusted Platform Module (TPM). Depending on the target platform on which the PQ measured boot will be integrated, the secure storage that will hold the measurements will be the TPM chip or an area of main memory isolated from the rest of the system via a Trusted Execution Environment (TEE), within which a firmware PQ-TPM can run. Then, the recorded quantum-safe measurements will be available to be used in the PQ Remote Attestation protocol.

### 3.3.3. Remote Attestation

Remote Attestation (RA) is a protocol used to verify the integrity and trustworthiness of a remote computing system. To enable it, the remote platform must provide mechanisms that allow a trusted party, often referred to as the *verifier*, to assess the actual state of the remote system, typically called *attester* or *prover*, in a way that is reliable even if the platform has been compromised.

RA is a challenge-response protocol in which the verifier sends a challenge (e.g., a nonce) to the attester, which then generates and signs an attestation evidence containing the requested measurements, the nonce, and other relevant data (e.g., timestamp, counters). The *nonce* is important in order to avoid *replay attacks*, in which the attester sends the verifier a non-fresh attestation evidence, created before the system is corrupted. In order to make the protocol resistant to replay attacks in a PQ scenario, the nonce has not to be predicted or forged even with the capabilities of a CRQC. Thus, the verifier has to generate the nonce by relying on a quantum-resistant Pseudo Random Number Generator (PRNG), based on hard problems believed to be resistant to quantum attacks, and with a length of at least 256 bits, given that CRQCs could potentially halve its security strength.

When the attesting system receives the attestation challenge from the verifier, it generates an attestation evidence, or *quote*, which serves as proof that the device's current state (including its firmware, software, and configuration) is trustworthy and has not been tampered with. The key elements of a quote are the measurements performed on the system components, the nonce received from the verifier and the signature made on the measurement data and the nonce to ensure integrity and authenticity. Therefore, in order to obtain a PQ quote, the measurement acquisition process must follow what is established in Section 3.3.2; the nonce must be generated by the verifier according to the guidelines described above; the signature must be done with a PQ digital signature algorithm. In order to enable runtime attestation of a computational node, the Integrity Measurement Architecture (IMA) module, provided by the Linux kernel, must be appropriately configured with a policy. To enable measurement of files accessed during system runtime with PQ-resistant hash algorithms, the IMA module has to be configured to use these algorithms (e.g., SHA-512, available starting from kernel version 3.13, SHA-3-512 available starting from version 6.7). Additionally, to protect the integrity of the IMA Log file by relying on SHA-512 or SHA-3-512, a TPM must be present on the system and it has to be configured with a Platform Configuration Register (PCR) bank associated with SHA-512 or SHA-3-512.

In the case of PQ RA, the selected signature algorithm should be efficient in all its operations, especially for signature generation and signature verification. The efficiency of these operations is crucial for quickly detecting compromises and maintaining overall system performance: the faster the attestation process, the shorter the window of opportunity for attackers to exploit a compromised system. Comparing the performance of the PQ algorithms proposed by NIST for standardization, FALCON [13] represents an optimal choice for PQ RA, because it provides the highest efficiency in signature generation and verification. ML-DSA [44] is also a suitable choice, having a good balance between key generation, signature generation, and signature verification efficiency, but is slightly slower than FALCON. Instead, if the maturity of the signature algorithm represents a preponderant requirement with respect to the performance of the remote attestation cycle, SLH-DSA is the algorithm to use. Since different evaluations can be made for RA based

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

on the use case, enabling algorithm agility for the attestation evidence signature algorithm is a desirable feature.

Finally, in order to protect the privacy of integrity measurements in transit on the network and to ensure that the attestation report is sent only to an authorized verifier, attester and verifier SHOULD establish a PQ secure communication channel, e.g., using PQ TLS, with PQ key exchange.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

# 4. IoT devices

## 4.1. Introduction

Resource-constrained IoT devices in the framework of post-quantum demand compact, low-power, and competitive solutions in terms of timing performance for the implementation of cryptographic functions.

Traditional hardware-assisted solutions based on TPMs are conceived for powerful processors included in high-end devices with high cost. However, mostly of the TPMs specifications are not suitable for IoT devices due to the limited resources of the platform. Furthermore, Hardware (HW) TPMs are not flexible, making the adoption of new cryptographic functionality difficult.

Secure Elements (SEs) emerge as alternative of the TPMs for the IoT devices. The integration of a SE in an IoT device enables robust hardware-based protection rather than just software security. The SE is purpose-built for IoT and meets the requirements with an affordable increase of price, when comparing with the unprotected version. Hardware SEs are widely used in various applications, such as Subscriber Identity Modules (SIMs), Europay, Mastercard, and Visa (EMV) bank cards, and electronic passports [45].

Current commercial SEs provide TPM-like functions, such as secure key generation, cryptographic operations, and identity generation. They rely on conventional asymmetric cryptography using RSA and ECC.

This section addresses the analysis and design of the different components needed to integrate PQC into IoT infrastructures. In particular, the transition exercise is focused on adopting a SE with PQ functionalities to IoT devices to harden infrastructure security against future quantum attacks.

QUBIP considers two different IoT design flavors:

- Micro-Controller Unit (MCU)-based IoT device: the SE is external to the IoT platform, therefore, a secure communication channel between MCU and the SE is established by means of a Secure Channel Protocol (SCP). The MCU leverages the SE to harden TLS implemented in Mbed-TLS (see Section 5.1).
- Micro-Processor Unit (MPU)-based IoT device: both the SE and MPU are implemented in a single System-on-Chip (SoC). In this more powerful IoT device, the OpenSSL cryptographic library can exploit HW implementations of PQ algorithms.

In addition, the SE can be used by the IoT device to perform cryptographic operations. The integration of the SE with the IoT device increases trust in the IoT ecosystem with an additional layer of security.

### 4.1.1. MCU-based IoT Device

The MCU-based IoT device represents a highly constrained system within the IoT domain, functioning on a microcontroller unit which operates within a bare-metal software environment or very reduced Operating System (OS). The MCU-based device will be able to connect to the Internet by using the provided on-board Ethernet port to reach a remote server that coordinates the action of multiple IoT devices with the same purpose. Additionally, the MCU-based IoT device is connected to one or more sensors that sample real-world data, and use the MCU to transfer the collected data to the remote server for analysis.

To improve the security of the embedded system, the MCU-based device establishes connectivity with a SE on-boarded on the same Printed Circuit Board (PCB), through the Inter Integrated Circuit (I2C) protocol. The MCU-based IoT device initiates calls to the SE to securely store or generate cryptographic material and to leverage the HW implementation of cryptographic algorithms.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

Two contexts are involved in the transition to PQC. First, in the communication protocol between the MCU-based device and the server, i.e., TLS 1.3 using PQ algorithms HW implementations in the SE. Second, in the serial communication protocol between the MCU-based device and the SE. In both contexts, the transition means a comprehensive adaptation to provide security against quantum attacks.

Internet communication will be based on TLS 1.3 [46]. We will be using the TLS implementation provided by the Mbed-TLS library to implement the PQ/T hybrid connections to the remote server. In order to reduce the load on the MCU-based device for supporting such a secure protocol, we will leverage the hardware implementations of the cryptographic primitives provided by the SE. Finally, in order to ensure the privacy and integrity of the sensitive data exchanged between the MCU and the SE, we will implement the SCP-03 on the MCU-SE I2C bus. Being based on symmetric cryptography, we need to select the security parameters compatible with the rest of the PQ-related design decisions.

### 4.1.2. MPU-based IoT Device

Embedded devices can come in various forms, depending on the needs of the application and its constraints, such as power consumption (e.g., battery-powered or energy harvesting devices), computational power, input/output and interfacing capabilities, cost, and security.

The choice of a particular device that manipulates data is usually tailored to a specific application (or at least a family of applications) or constraints. In some applications, MCUs are not the optimal solutions due to their limitations in terms of computational power, at the cost of higher power consumption and more expensive hardware. Therefore, when complex tasks must be performed and/or data manipulation requires tighter timing requirements, the solution is to adopt MPU-based systems. MPUs-based IoT devices share many things in common with MCU-based devices, but they are generally more advanced and complex. From a processing viewpoint, MPUs-based devices are usually faster than MCU-based devices and are generally capable of running a full-featured OS. The possibility to leverage an OS is not always feasible on MCU-based IoT devices[1] due to limited capabilities, but it enables MPU-based systems to perform very complex tasks and to ease the development. In addition, MCUs' memory is usually limited, and the code's size represents a huge limitation for writing complex applications. From a connection standpoint, MPUs usually come with many peripherals that are similar to MCUs', but they also have the possibility to connect to high-speed communication peripherals, such as USB-3.0 and/or Gigabit Ethernet ports, as their computational power and memory can support large amounts of data. Memory-wise, it is common for MPUs-based systems to store their code outside the device itself, and external memory for storage and Random Access Memory (RAM) is usually needed, which makes in general the Bill-Of-Materials (BOM) more expensive with respect to an MCU-based system.

In advanced IoT scenarios, an MPU-based device establishes a TLS secure channel with a server to exchange data collected from other IoT nodes and/or sensors. Moreover, the Software (SW) Integrity Verification at boot and runtime is crucial and implemented by means of the Trusted Computing (TC) techniques described in Chapter 3.

The transition to PQC involves the implementation PQ algorithms to support PQ/T TLS by means of HW acceleration and PQ integrity verification. In addition, the transition of the integrity verification techniques requires a PQ firmware Trusted Platform Module (fTPM) running into a TEE.

#### Securing TLS with PQC

Speeding-up the TLS communication between the IoT endpoint and a server is crucial, especially when dealing with transition to PQC. The possibility to speed up PQ algorithms execution can be achieved by ensuring that these “heavy” functions are all performed in hardware. In this case, the “heavy” functions

[1]Some MCUs are able to run some lightweight OSs and real-time OSs.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

are not performed by the Central Processing Unit (CPU) itself, but on a dedicated area of silicon that has been specifically designed to perform such operations. Clearly, a possible solution could be to leverage on off-chip SEs, as described in Section 4.1.1.

On the other hand, MPUs-based devices are commonly organized as a SoCs, where a powerful CPU running a rich OS is coupled with on-chip hardware accelerators, aiming at off-loading the CPU itself from heavy computational tasks. Typically, the on-chip communication between the CPU and the hardware accelerators is made possible by means of standard interfaces such as Advanced eXtensible Interface (AXI) and Advanced Peripheral Bus (APB). As PQ algorithms typically require a large number of complex operations, it is possible to implement directly on-chip the logic to speed-up cryptographic tasks for PQ and PQ/T hybrid algorithms by means of adopting the hardware acceleration strategy. The transition to PQC strongly benefits from the large bandwidth provided by on-chip interfaces compared to the off-chip SE solution.

The hardware approach is also followed by the adoption of a proper OpenSSL version with PQ and PQ/T hybrid TLS protocols implementation. Hence, OpenSSL can exploit the benefit of hardware acceleration through a set of custom drivers.

#### PQ Root of Trust for Integrity Verification

As stated in Chapter 3, the transition of TC procedures to PQC is paramount to ensure quantum-secure integrity verification in embedded systems. A key-enabler technology to achieve TC is represented by the Root of Trust (RoT), which is a component, either made in hardware or in software, that provides a firm foundation to build security and trust in a system. A classical example of RoT is represented by TPMs, that offer a secure enclave with physical and/or logical separation for storing sensitive materials and to perform cryptographic operations.

Our transition exercise clearly requires a PQ RoT to provide quantum-secure TC. It has be to noted that commercial hardware TPMs (usually a standalone device) do not support PQ functionalities yet, and therefore they are not suitable to be a PQ RoT. In modern MPUs, it is possible to leverage on TEEs already at CPU-level to provide TPM services. In such TEEs, the isolation required to provide TPM functionalities is typically provided by the internal logic of the CPU and by the OS running on it. This strategy allows the implementation of the needed TPM functionalities in firmware, *de facto* delivering the possibility to implement a fTPM. The adoption of the fTPM is a suitable way to enable a quantum-secure integrity verification for an MPU-based IoT device, allowing secure and measured boot as well as RA.

### 4.1.3. Secure Element

Current IoT systems establish trusted connections among their components. These systems require a trust anchor, which is a secure location for storing essential secrets or performing secure computations. It is widely recognized that it is inherently impossible to create such a trust anchor using only software. For that reason, hardware SEs have been developed and are now increasingly prevalent [47].

The SEs play a crucial role in ensuring the security by implementing cryptographic algorithms linked with symmetric or asymmetric keys. In addition, they offer a variety of key management mechanisms, including secure storage and key generation. It is also essential that these elements are protected through a combination of logical and physical security precautions [48].

The security requirements for SEs are dictated by the FIPS 140-3 standard [49], which replaces the previous FIPS 140-2 standard [50], also referred to as “Security Requirements for Cryptographic Modules”. This standard establishes a range of escalating security levels from 1 to 4.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

## 4.2. Requirements

The requirements for each component of the IoT devices are detailed in Tables 4.1, 4.2, and 4.3. The analysis distinguishes the requirements of the MCU-based IoT device (see Table 4.1), MPU-based IoT device (see Table 4.2), and SE (see Table 4.3), including a short and concise description.

**Table 4.1:** MCU-based IoT device requirements

| Req. | Description |
|---|---|
| MCU-01 | PQ/T hybrid TLS implementation MUST leverage PQ algorithms selected by NIST. New algorithms can be considered as the NIST standardization process evolves |
| MCU-02 | The PQ/T hybrid TLS implementation MUST use PQ algorithms that provide the best trade-offs for efficiency, power consumption, memory occupancy and implementation area |
| MCU-03 | The communication between the MCU and the SE MUST involve symmetric algorithms for encryption and authentication which are resistant to quantum threats |

**Table 4.2:** MPU-based IoT device requirements

| Req. | Description |
|---|---|
| MPU-01 | The PQ/T hybrid key exchange mechanism MUST use PQ KEM algorithms selected by the NIST. New PQ KEMs can be considered as the NIST standardization process evolves |
| MPU-02 | The PQ/T hybrid signature generation and verification mechanisms MUST use PQ signature algorithms selected by NIST. New algorithms can be considered as the NIST standardization process evolves |

**Table 4.3:** SE requirements

| Req. | Description |
|---|---|
| SE-01 | The key exchange mechanism implemented in the SE MUST support the use of classical and PQ solutions. It MUST be compliant with the selected algorithms of the MCU and MPU-based devices |
| SE-02 | The SE MUST provide implementation of PQ/T hybrid signature algorithms. It MUST be compliant with the selected algorithms of the MCU and MPU-based devices |
| SE-03 | An Application Programming Interface (API) based on the standard PKCS#11 MUST be provided to ease the manipulation of the SE |
| SE-04 | The communication between the MCU-based device and the SE MUST be secured through the use of a quantum-secure serial communication protocol |

On the MPU board, integrity verification can be implemented through Secure Boot, Measured Boot and Remote Attestation, following the requirements of Table 3.1. The transition to PQC exercise will be performed by integrating PQ digital signature algorithms at firmware level during Secure Boot and strong hash algorithms, resistant to quantum computers, during Measured Boot, as explained in Chapter 3.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

## 4.3. Design

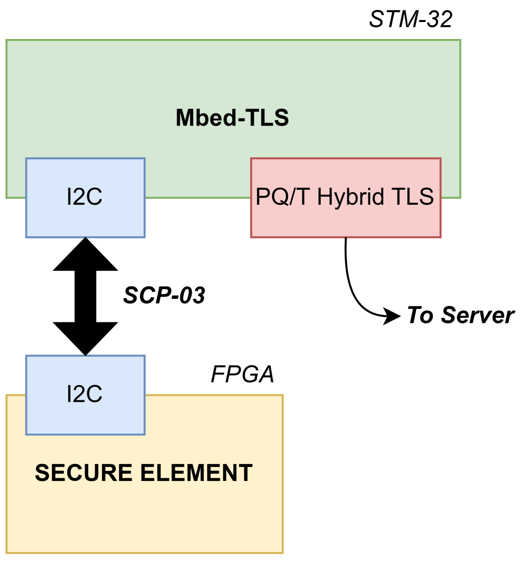
After defining the requirements, the next step involves the design of these components. The device comes in two variants, MCU-based and MPU-based, as shown in Figures 4.1a and 4.1b, respectively.

The MCU-based variant consists of two distinct parts: an STM-32 running Mbed-TLS and an SE linked to the STM-32 via I2C interface and protocol. To guarantee secure communication between these components, the SCP-03 protocol [51] is used. Our design makes use of larger challenge size, from 8 to 16 bits (from S8 to S16 [51]) to avoid possible attacks based on Grover's algorithm [41], due to the low entropy of nonces. In addition, the TLS protocol in Mbed-TLS leverages the HW implementation of the PQ/T hybrid algorithms provided by the SE.

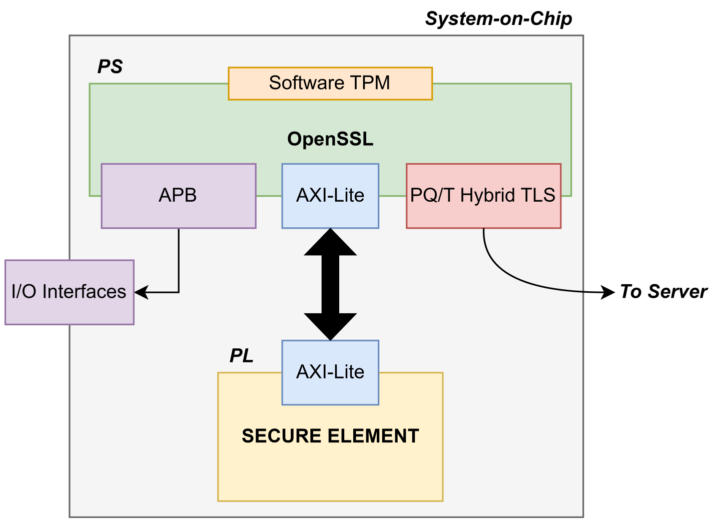
The developing board selected for the implementation of the MCU-based IoT devices is the nucleo-144 STM32F4 [52]. It is one of the most widely used and deployed MCU-based IoT device on the market, offering good computational capabilities and low power consumption. Furthermore, it comes with a series of hardware accelerators on some models, and offers a wide range of peripherals. The Field Programmable Gate Array (FPGA) chosen for the implementation of the SE is the Genesys-2 board [53], which features a Kintex-7 FPGA (XC7K325T-2FFG900C). This FPGA, with its 478K LUTs, is capable of implementing all the cryptographic primitives required by the SE and offering at the same time a beneficial balance between cost and performance.

The MPU-based variant is developed on a SoC. In this setup, OpenSSL operates on an OS whose Processing System (PS) is based on an ARM core, adhering to the requirements outlined in Tables 4.2. The SE, which meets the requirements listed in Table 4.3, is implemented in the Programmable Logic (PL) and connects to the ARM core via the AXI-Lite protocol [54]. The ARM core manages the connection to the server using the PQ/T hybrid TLS implementation with assistance from the SE. The APB is utilized for external connections to other devices or services. The development board chosen for the implementation is the ZCU-104 [55]. This board features a Zynq UltraScale+ (XCZU7EV-2FFVC1156 MPSoC) with a PS that is an ARM Cortex-A53 and a PL with over 500K logic cells. The choice was influenced by the experience of the consortium with the Pynq platform, which can be implemented on this board, easing the hardware integration as well as the resource requirements for the final design of the SE.

At firmware level, the First Stage Bootloader (FSBL) and the ARM Trusted Firmware (ATF) [56] compo-



(a) MCU-based design



(b) MPU-based design

**Figure 4.1:** Two different design flavours of the IoT device.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

nents, involved in the booting process of the MPU, can be modified in order to have Secure Boot and support PQ algorithms.

For measured boot and remote attestation, a fTPM with integration of PQ algorithms will be implemented inside ARM TrustZone as a Trusted Application using the Open Portable Trusted Execution Environment (OP-TEE) framework [57].

QUBIP is going to develop a single SE and use it in the two design variants. The choice of the algorithms to be implemented in the SE plays a crucial role. In order to meet the requirements set by the different components (i.e., TLS, SCP, Integrity Verification) running in the two IoT device flavors, and with the idea of supporting a PQ/T hybrid approach where relevant, the SE provides the implementation of the following algorithms:

- **Hash**: SHA-256, SHA-384, SHA-512, SHA-512/256, SHA3-256, SHA3-512, SHAKE-128, SHAKE-256.
- **Symmetric encryption**: AES-128-ECB, AES-128-CBC, AES-128-GCM, AES-128-CMAC, AES-256-ECB, AES-256-CBC, AES-256-GCM, AES-256-CMAC.
- **KEM**: X25519 (classical) and ML-KEM-512/764/1024 (PQC).
- **Digital signature**: EdDSA25519 (classical), ML-DSA-44/65/87 (PQC) and SLH-DSA-SHA2/SHAKE-128/192/256-X (PQC) (not yet selected the 's' or 'f' version).
- **Key Deviation Function**: HKDF-SHA256.
- **Random Number/Bit Generators**: TRNG and DRBG-AES/SHA2.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

# 5. Cryptographic Libraries

## 5.1. Introduction

Cryptographic libraries are essential tools in modern software development, providing a suite of algorithms and schemes designed to secure data through encryption, decryption, hashing, and authentication. These libraries offer developers a robust framework for implementing complex cryptographic functions without requiring in-depth knowledge of the underlying mathematics. Some cryptographic libraries further provide support for secure communication protocols, with varying degrees of abstraction and tuneable parameters. Notable examples, such as OpenSSL, demonstrate the critical role of open-source cryptographic libraries in ensuring secure communications and data integrity across diverse platforms.

In the rest of this chapter we provide a brief introduction to the building blocks involved in the “Cryptographic libraries” layer of the multilayered approach of the QUBIP project for the transition to PQC. In particular, we target OpenSSL, Network Security Services (NSS), and Mbed-TLS, to seamlessly provide support for PQ primitives and schemes, and PQ/T Hybrid TLS, as detailed below.

After this introduction, Section 5.2 covers the requirements of the building blocks in this layer, while Section 5.3 covers the matching design specifications.

### OpenSSL

OpenSSL [58] is a a robust, commercial-grade, full-featured open-source toolkit for general-purpose cryptography and secure communication. Originating in 1998, it serves as a crucial component in securing communications over computer networks, enabling encrypted data transfer and authentication. It consists of three main components, namely:

**`libcrypto`** a library offering access to cryptographic primitives and cross-platform abstractions;

**`libssl`** a library mainly implementing support for the TLS protocol;

**built-in applications** a collection of command-line utilities to create and manipulate various cryptographic objects, and test supported communication protocols.

As an open-source project, OpenSSL benefits from a collaborative development model, allowing continuous improvements and community-driven support. OpenSSL’s extensive range of cryptographic algorithms and protocols makes it an essential tool for developers and organizations aiming to enhance the security of their digital infrastructure. For all these reasons, OpenSSL is widely pervasive, especially on Linux-based systems, forming a foundational element in a multitude of applications and services, with its codebase comprising millions of lines of code.

As part of OpenSSL 3.0, the latest major release of the project, significant efforts have been devoted to developing a new software architecture. This architecture is characterized by the introduction of `Providers`, which serve as containers for implementations of cryptographic operations [59]. This new architecture is mainly aimed at enhancing cryptographic agility, ensuring that applications utilizing high-level APIs interact seamlessly with the OpenSSL `core` layer. This `core` layer dynamically dispatches requests to the preferred `Provider` implementation at run-time. The `Provider` architecture’s extensibility allows users to configure their systems or applications to load third-party `Providers`. These third-party `Providers` can offer access to cryptographic schemes not yet available in the main OpenSSL distribution or provide implementations that exploit the specific capabilities of the current system, integrating them into the network protocol stack.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

In QUBIP we leverage the extensibility of the `Provider` architecture, to achieve a seamless transition to PQC. We generate customized *shallow* `Providers`, configured to expose the choice of PQ schemes designated by the developers, with their choice of underlying implementations, and integrate them in PQ/T Hybrid [60] schemes for TLS 1.3 [46].

#### NSS

NSS [61] is a suite of libraries crafted to facilitate the development of security-enabled client and server applications across various platforms, leveraging the Netscape Portable Runtime (NSPR) project [62], a platform-neutral open-source API designed to facilitate cross-platform development. NSS provides support for, among others, TLS 1.2 [63], TLS 1.3 [46], Secure/Multipurpose Internet Mail Extensions (S/MIME) [64], X.509 v3 certificates, Public-Key Cryptography Standards (PKCS) #8 [65], PKCS #12 [66], and numerous other security standards. While Mozilla hosts its source code repository and the issue tracker and infrastructure to support its development process, NSS is co-developed by various companies and individual contributors, including Red Hat, Oracle, and Google. Albeit being integrated in many open-source client and server solutions, in the context of the QUBIP project, we mostly focus on the use of NSS within the Mozilla Firefox Internet browser.

NSS natively supports PKCS #11 [67], to access HW and SW implementations of cryptographic primitives. Within QUBIP we aim to leverage the PKCS #11 support to implement shallow dynamically loadable modules, akin to the discussed QUBIP OpenSSL `Providers`, to allow developers and users to inject support for PQC schemes in NSS, making them available to NSS-powered applications.

#### Mbed-TLS

Mbed-TLS [68], originally known as PolarSSL, was developed to provide an easy-to-use and high-performance SSL/TLS library. ARM Holdings acquired PolarSSL in 2014 and rebranded it as Mbed-TLS, integrating it into the Mbed platform to enhance IoT security. The library is currently maintained and developed by the Trusted Firmware Project [69]. Mbed-TLS is written in C language and implements essential cryptographic primitives, which are the foundational algorithms and protocols for securing data. These include symmetric key algorithms such as AES, hashing functions such as SHA-256, and asymmetric key algorithms such as ECC.

In addition to these primitives, Mbed-TLS offers extensive support for X.509 certificate manipulation. The library provides functionalities for parsing, creating, and verifying these certificates, enabling secure identity verification and data encryption.

Moreover, Mbed-TLS includes implementations of the TLS and Datagram Transport Layer Security (DTLS) protocols. These protocols are designed to establish secure communication channels over the Internet.

Mbed-TLS has been specifically designed for providing cryptographic functionalities having a small code footprint. This characteristic makes the library particularly well-suited for use in embedded systems, where memory and storage resources are often limited. For example, Mbed-TLS is used extensively in Trusted Firmware-A (TF-A) [70], Trusted Firmware-M (TF-M) [56], and OP-TEE [57].

## 5.2. Requirements

In this section, we collect the requirements for the building blocks of the cryptographic library layer. This is arranged with a separate subsection for each of the building blocks.

Notice that among the listed requirements, we deliberately included some functional requirements that are not directly related with the PQ transition exercise, but are more generic. We included such requirements

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

in this document only when we deemed it necessary, as those constraints either still informed PQ transition design decisions detailed here, or because they further impose requirements on the dependencies selected as part of the specific demonstrators.

### 5.2.1. OpenSSL Providers: Requirements

**Table 5.1:** OpenSSL Providers requirements

| Req. | Description |
|---|---|
| OSSLProv-01 | The integration between QUBIP `Provider` and OpenSSL MUST allow OpenSSL (and applications above) to seamlessly select the correct primitives to handle new abstract cryptographic objects (i.e., keys, certs, etc.) |
| OSSLProv-02 | The integration between QUBIP `Provider` and OpenSSL SHOULD ensure that the integration in applications above is as transparent as allowed by OpenSSL |
| OSSLProv-03 | The QUBIP `Provider` MUST enable PQ/T Hybrid Key Encapsulation Method (KEM) in TLS 1.3 handshakes |
| OSSLProv-04 | The QUBIP `Provider` MUST enable PQ/T Hybrid Signatures in TLS 1.3 handshakes |
| OSSLProv-05 | Once standardized, the QUBIP `Provider` MUST provide full support for PQ/T Hybrid PKI (i.e., not only the handshake signatures, but PQ/T Hybrid certificate chains up to the root of trust) |
| OSSLProv-06 | The QUBIP `Provider` MUST enable full support for PQ/T Hybrid server-side authentication in TLS 1.3 |
| OSSLProv-07 | The QUBIP `Provider` MUST enable full support for PQ/T Hybrid client-side authentication in TLS 1.3 |
| OSSLProv-08 | The QUBIP `Provider` SHOULD expose a number of PQ/T Hybrid schemes, for both key exchange and authentication in TLS 1.3, in different combinations at different security levels |
| OSSLProv-09 | The QUBIP `Provider` SHOULD expose the PQC and PQ/T Hybrid primitives for all other uses of OpenSSL, not just enable PQ TLS 1.3 |
| OSSLProv-10 | The QUBIP `Provider` MUST be released under an open-source license compatible with OpenSSL and with the licensing restrictions of external implementations |
| OSSLProv-11 | The QUBIP `Provider` MUST NOT contain crypto implementations inside of it, but rather link/incorporate implementations coming from external projects |
| OSSLProv-12 | The QUBIP `Provider` SHOULD be implemented using State of the Art (SotA) memory safety practices (e.g., Rust or other technologies with similar higher assurance on memory safety) |
| OSSLProv-13 | The QUBIP `Provider` SHOULD be compiled as a dynamically loadable module |
| OSSLProv-14 | The design SHALL allow to select external software projects as the back-end implementation of the shallow module low level primitives |
| OSSLProv-15 | The main development and testing platform for the QUBIP `Provider` and the OpenSSL integrations SHALL be Linux |

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

| | |
|---|---|
| OSSLProv-16 | At build-time, the design SHOULD generate as much scaffolding as possible from a declarative description of the algorithms, PQ/T Hybrids, and back-end implementations |
| OSSLProv-17 | The QUBIP `Provider` SHOULD be functionally compatible with the `oqsprovider` from the OQS project |
| OSSLProv-18 | Already during development, the QUBIP `Provider` SHOULD strive to ensure interoperability with other PQ/T Hybrid TLS 1.3 stacks. The OQS project is the primary interoperability target, but cross-testing with other stacks MAY be pursued |

### 5.2.2. NSS: Requirements

**Table 5.2:** NSS requirements

| Req. | Description |
|---|---|
| NSS-01 | The integration between QUBIP `Module` and NSS MUST allow NSS (and applications above) to seamlessly select the correct primitives to handle new abstract cryptographic objects (i.e., keys, certs, etc.) |
| NSS-02 | The integration between QUBIP `Module` and NSS SHOULD ensure that the integration in applications above is as transparent as allowed by NSS |
| NSS-03 | The integration between QUBIP `Module` and NSS SHOULD allow NSS to dynamically discover supported low level PQC primitives and PQ/T Hybrid from the loaded QUBIP `Module`, and expose them for use at the protocol and the application layers |

### 5.2.3. NSS Modules: Requirements

**Table 5.3:** NSS Modules requirements

| Req. | Description |
|---|---|
| NSSMod-01 | The QUBIP `Module` MUST enable PQ/T Hybrid KEM in TLS 1.3 handshakes |
| NSSMod-02 | The QUBIP `Module` MUST enable PQ/T Hybrid Signatures in TLS 1.3 handshakes |
| NSSMod-03 | Once standardized, the QUBIP `Module` MUST provide full support for PQ/T Hybrid PKI (i.e., not only the handshake signatures, but PQ/T Hybrid certificate chains up to the root of trust) |
| NSSMod-04 | The QUBIP `Module` MUST enable full support for PQ/T Hybrid server-side authentication in TLS 1.3 |
| NSSMod-05 | The QUBIP `Module` MUST enable full support for PQ/T Hybrid client-side authentication in TLS 1.3 |
| NSSMod-06 | The QUBIP `Module` SHOULD expose a number of PQ/T Hybrid schemes, for both key exchange and authentication in TLS 1.3, in different combinations at different security levels |

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

| | |
|---|---|
| NSSMod-07 | The QUBIP `Module` SHOULD expose the PQC and PQ/T Hybrid primitives for all other uses of NSS, not just enable PQ TLS 1.3 |
| NSSMod-08 | The QUBIP `Module` MUST be released under an open-source license compatible with NSS and with the licensing restrictions of external implementations |
| NSSMod-09 | The QUBIP `Module` MUST NOT contain crypto implementations inside of it, but rather link/incorporate implementations coming from external projects |
| NSSMod-10 | The QUBIP `Module` SHOULD be implemented using SotA memory safety practices (e.g., Rust or other technologies with similar higher assurance on memory safety) |
| NSSMod-11 | The QUBIP `Module` SHOULD be compiled as a dynamically loadable module |
| NSSMod-12 | The main development and testing platform for the QUBIP `Module` and the NSS integrations SHALL be Linux |
| NSSMod-13 | Already during development, the QUBIP `Module` SHOULD strive to ensure interoperability with other PQ/T Hybrid stacks. The OQS project is the primary interoperability target, but cross-testing with other stacks MAY be pursued |

### 5.2.4. Mbed-TLS: Requirements

**Table 5.4:** Mbed-TLS requirements

| Req. | Description |
|---|---|
| MbedTLS-01 | The integration of PQC algorithm implementations in Mbed-TLS MUST allow Mbed-TLS (and applications above) to seamlessly select the correct primitives to handle new abstract cryptographic objects (i.e., keys, certificates, etc.) |
| MbedTLS-02 | The integration of PQC algorithm implementations in Mbed-TLS SHOULD ensure that the integration in applications above is as transparent as allowed by Mbed-TLS |
| MbedTLS-03 | Mbed-TLS MUST ensure the ability to seamlessly select the classical primitives provided by the HW SE |
| MbedTLS-04 | Once standardized, the Mbed-TLS MUST provide full support for PQ/T Hybrid PKI (i.e., not only the handshake signatures, but PQ/T Hybrid certificate chains up to the root of trust) |

## 5.3. Design

Based on the requirements listed in the previous section, here we describe our design in terms of specifications and rationale.

### 5.3.1. OpenSSL Providers: Design

We developed a design out of the requirements listed in Table 5.1. The design adopts the OpenSSL `Provider` API [59] for the integration of the QUBIP `Provider` and OpenSSL. The `Provider` API is the recommended mechanism in the latest OpenSSL releases to perform this kind of seamless integration. Modern applications built on OpenSSL usually delegate the selection of the correct primitives to handle the new abstract cryptographic objects to the OpenSSL `core`, which can be influenced via a configuration file or Command Line Interface (CLI), without further changes to existing applications. The `Provider` API natively supports building `Providers` as dynamically loadable modules: by limiting our design to use exclusively features supported by the `Provider` API, we can thus be compliant with both OSSLProv-01 and OSSLProv-13 requirements.

**Integration with OpenSSL's High-level APIs.** Applications rigorously using only the high level APIs offered by OpenSSL, i.e., `EVP` API and `libssl` functions, will seamlessly benefit from the integration through the `Provider` API, achieving compliance with OSSLProv-02. Applications that have not fully completed the migration to OpenSSL `3.0` and those that still use deprecated APIs, cannot be fully reached by the new OpenSSL `core` architecture and the `Provider` API: to fully achieve OSSLProv-02 our designs requires relevant applications to be following OpenSSL `3.0` recommendations.

**Support for KEM and `Signature` Operations.** The `Provider` API natively provides support for generic KEM as well as generic `Signature` operations, and, as of OpenSSL `3.2` a *capabilities* mechanism to inject support for both operations in the TLS 1.3 stack of `libssl` [71]. Our design leverages this feature of the `Provider` API to achieve OSSLProv-03 and OSSLProv-04, using combiners to expose generated PQ/T Hybrid schemes for both operations.

**Integration of PQ/T Hybrid `Signature` Into PKI.** Similarly, the new OpenSSL `core` architecture and the `Provider` API, allow for arbitrary pluggable `Signature` operations into the PKI stack of the library. Our design leverages this feature of the `Provider` API for compliance with OSSLProv-05. We will continue to closely monitor the ongoing discussions about the standardization of PQ/T Hybrid schemes for Signatures (e.g., see [30]) to ensure the QUBIP `Provider` will support suitable *combiners*.

**Cryptographic Agility and Evolving Standards.** On a related note, our design relies on continuous monitoring of standardization activities around *combiners* for all relevant PQ/T Hybrid schemes, to ensure suitable schemes for both key exchange and authentication can be implemented in TLS 1.3, with the required combinations and at different security levels as required by the use cases in accordance with standardization actions. We are confident that the degree of cryptographic agility offered by the new OpenSSL `core` architecture will allow us to conform to OSSLProv-08, and we plan to assist the upstream development of OpenSSL to pursue the changes that might be needed to fully support PQ/T Hybrid PKI.

**TLS 1.3 Authentication.** The design of the QUBIP `Provider` leverages the `Provider` API to provide PQ/T Hybrid authentication support in TLS 1.3 both on server- and client-side, addressing OSSLProv-06 and OSSLProv-07. This feature is already supported upstream since OpenSSL `3.2`, but as noted before, further patches upstream might be required by the advance of standardization of PQ/T Hybrid `Signature`.

**Access to PQC Primitives.** The design of the QUBIP `Provider` not only enables PQ TLS 1.3 but also exposes PQC primitives and PQ/T Hybrid schemes for all other uses of OpenSSL. This is achieved leveraging the already supported features of the `Provider` API, in compliance with OSSLProv-09. Our design involves comprehensive integration and compatibility testing using OpenSSL APIs.

**Shallow Module Methodology.** Our design for the QUBIP `Provider` is based on our *shallow module* methodology, i.e., it does not include cryptographic implementations inside of it, but rather links (or otherwise incorporates) PQ implementations from external projects, fulfilling OSSLProv-11. This is fully

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

supported by the `Provider` API, and our design will further facilitate the integration of external PQ implementation projects, as long as the back-end implementations expose an API compatible with the API [72] published by National Institute of Standards and Technology (NIST) as part of the PQC standardization process [8].

**Licensing.** The QUBIP `Provider` will be released under the same license as OpenSSL, i.e., Apache License v2 [73]. To fulfill OSSLProv-10, each use-case selection criteria for the back-end implementations of PQ primitives, which are provided via external projects, MUST include considerations on licensing terms and ensure license compatibility with Apache License v2 [73].

**Compatibility with External Primitive Implementations.** Furthermore, the above design decisions inform the design to fulfill OSSLProv-14. In particular, the partners selecting back-end external implementations to be integrated with the QUBIP `Provider`, must ensure that the selected projects

1. conform to the official NIST PQC API notes [72];
2. are released under licensing terms that ensure compatibility with Apache License v2 [73].

Our design involves comprehensive integration and compatibility testing against the implementations collected by the OQS project, to ensure that the selected back-ends are functionally correct, which partially addresses OSSLProv-18.

**Functional Compatibility target.** The overall design also includes functional compatibility testing against the `oqsprovider`, in fulfillment of OSSLProv-17. This in practice means that applications tested using the `oqsprovider` should expect to function identically using our QUBIP `Provider`. Implicitly this also ensures that, at the protocol level, our design will continuously ensure interoperability with other selected PQ/T Hybrid stacks and other protocol implementations, fulfilling OSSLProv-18. We plan to cover at least the same interoperability targets selected by the maintainers of `oqsprovider`, and possibly add additional ones as needed for the specific use-case demonstrators.

**Secure Coding Practices.** The design of the QUBIP `Provider` will use SotA memory safety practices, using `Rust`, in fulfillment of OSSLProv-12. By adopting best `Rust` practices, we will leverage its memory- and thread-safety assurances, lowering the risks of introducing security critical bugs. We note that the OpenSSL `Provider` API mandates a `C` Application Binary Interface (ABI), which although supported by `Rust` requires the use of `C` Foreign Function Interface (FFI) bindings, which in general require `unsafe` code blocks. Within such blocks many of the higher assurances of the `Rust` language are not guaranteed by the `Rust` tool chain, and in our design we plan to implement idiomatic `Rust` best practices, isolating `unsafe` blocks in the smallest possible units, to aid thorough manual assessment. The `unsafe` FFI bindings will be further encapsulated within convenience `Rust` functions and methods, which will offer an idiomatic way to interfacing with the FFI bindings after ensuring the required conditions for safe use.

**Scaffolding Nature of the Design.** Our design further leverages the capabilities of the `Rust` language also to generate as much scaffolding as possible from a declarative description of the algorithms, hybrids, and back-end implementations. This will be achieved primarily through idiomatic `Rust` features, such as `Macros` [74, Chapter 19.5 "Macros"], automating code generation processes based on predefined templates and configurations to avoid repetitive work and hopefully bugs, in fulfillment of OSSLProv-16.

**Designated Development Platform.** Finally, we will focus on Linux as the main development and testing platform for the QUBIP `Provider`, in accordance with OSSLProv-15. In particular, we selected

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

Fedora Linux as the primary testing platform, based on the fact that this will be done in collaboration with Red Hat, which is targeting Fedora Linux in their tasks. We note that, although support for other platforms such as MacOS and MS Windows (as long as they are supported by the selected back-end projects) is desirable, it is considered a non-requirement for the scope of the QUBIP `Provider`. Nonetheless, while we won't guarantee portability to other platforms, our design strives to pursue best practices in terms of portability, to facilitate future porting efforts.

### 5.3.2. NSS: Design

The NSS PQC transition will be facilitated through the use of loadable modules. Loadable modules in our design refer to dynamically loadable shared libraries that extend the capabilities of NSS without requiring a recompilation of the NSS library itself. PKCS #11, also known as the Cryptographic Token Interface Standard, defines a platform-independent API to manage and use cryptographic tokens [67]. NSS uses the PKCS #11 to interface with cryptographic hardware and software tokens. By adhering to the PKCS #11 standard, NSS can dynamically load and utilize cryptographic modules, making it possible to seamlessly integrate new cryptographic algorithms and functionalities. Modifications to the NSS source code are intended to be minimal. These changes primarily involve configurations to enable NSS to recognize the module and TLS 1.3 to recognize the PQC functionality within the protocol. This approach ensures that our design remains easily maintainable.

**Integration of the QUBIP `Module` and NSS.** Our design meets the requirements in the Table 5.2 by ensuring the seamless integration between the QUBIP `Module` and NSS. The QUBIP `Module` implementation of `Cryptoki` API and NSS's native support for PKCS #11 will allow for seamless integration between NSS and the QUBIP `Module`. This integration enables the correct selection of primitives and handling of new cryptographic objects, ensuring compliance with NSS-01.

**Cryptographic Agility.** To achieve compliance with NSS-02, our design targets the seamless integration of applications within the constraints posed by NSS. We plan to extend the current capabilities of NSS, to further the cryptographic agility of the interface between NSS and applications built on top of it. Thus, our design also anticipates future integration needs that may overcome the current constraints.

**Discovery of Primitives.** To achieve NSS-03, the design will proceed in several iterations. Initially, due to constraints in NSS, we will hardcode the discovery of supported low-level PQC and PQ/T Hybrid primitives, and the related TLS 1.3 codepoints. This approach will pave the way for the project's targeted phase: the dynamic discovery of low-level primitives for use at the protocol and application layers, similar to what is currently achievable in Section 5.3.1 via the `Provider` API.

### 5.3.3. NSS Modules: Design

The QUBIP `Module` is the primary method for integrating PQC functionality into Mozilla Firefox through NSS. Its core design principle is its shallow nature, meaning the module avoids nested or deeply layered components. This design makes the module easy to understand, maintain, and integrate. By adopting this approach, we can produce a QUBIP `Module` with high cryptographic agility, serving as a framework that allows other researchers to test their cryptographic components in a web browsing setting with minimal effort.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

**Components.** The design of the QUBIP `Module` leverages NSS's native support for PKCS #11 [67], enabling us to structure the QUBIP `Module` with the following components:

1. `Cryptoki` API,
2. a software interoperability layer,
3. back-end cryptographic implementations.

**`Cryptoki` API.** The `Cryptoki` API, specified by the PKCS #11 standard, provides applications with a common logical view of a device capable of performing cryptographic functions, known as a cryptographic token. The PKCS #11 standard defines the data types and functions available to applications requiring cryptographic services using header files in the ANSI C programming language. Consequently, an application only needs to use the functions specified in the PKCS #11 standard to access the cryptographic functionality provided by the cryptographic token. From the NSS's perspective, a cryptographic token is the QUBIP `Module` that acts as a software token, providing PQC functionality. Additionally, the design includes a software interoperability layer that connects the `Cryptoki` API and the back-end implementation into a functional entity. This ensures seamless integration and interaction between the API and the back-end implementation.

**Support for KEM and `Signature` Operations.** The QUBIP `Module` will support PQ/T Hybrid KEM and `Signature` operations, integrated with the TLS 1.3 stack of NSS, thus fulfilling NSSMod-01 and NSSMod-02. The current NSS architecture is limited in its support for arbitrary pluggable `Signature` operations within the PKI stack of NSS. Therefore, fully addressing NSSMod-03 requires advancing to the final iteration of the design addressing NSS-03. As mentioned in Section 5.3.2, in the initial design iteration we will hardcode the discovery of low level primitives offered by the QUBIP `Module`; our design aims to later enhance the dynamic discoverability of the QUBIP `Module` capabilities, within the constraints of the `Cryptoki` API. We are also closely following the ongoing discussions on the standardization of PQ/T Hybrid schemes for `Signature` (e.g., see [30]) to ensure the QUBIP `Module` will support suitable *combiners*.

**TLS 1.3 Authentication.** The design of the QUBIP `Module` leverages the `Cryptoki` API to provide PQ/T Hybrid authentication support in TLS 1.3 for both server-side and client-side, addressing NSSMod-04 and NSSMod-05.

**Cryptographic Agility and Evolving Standards.** Similar to how Section 5.3.1 addresses OSSLProv-08, we are monitoring the standardization around *combiners* for all relevant PQ/T Hybrid schemes to ensure suitable key exchange and authentication in TLS 1.3. With the cryptographic agility and flexibility our approach offers, we are confident that we can conform to NSSMod-06.

**Licensing.** The QUBIP `Module` exposes its implementation of the `Cryptoki` API, as well as the PQC primitives and PQ/T Hybrid schemes for all other uses of NSS, thus complying with NSSMod-07. NSS is released under the Mozilla Public License Version 2.0 [75]. We will release our QUBIP `Module` under the same license, adhering to NSSMod-08.

**Shallow Module Methodology.** The design for the QUBIP `Module` is based on our *shallow module* methodology. i.e., it does not include cryptographic implementations inside of it, but rather links (or otherwise incorporates) PQ implementations from external projects, fulfilling NSSMod-09. Similarly to the

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

design decision for OSSLProv-10, we advise that, to fulfill NSSMod-08, each use-case selection criteria for the back-end implementations of PQ primitives, which are provided via external projects, MUST include considerations on licensing terms and ensure license compatibility with Mozilla Public License Version 2.0 [75].

**Secure Coding Practices.** The QUBIP `Module` will be developed in the `Rust` programming language. This approach aligns with Mozilla's Oxidation project [76], which strongly advocates for using `Rust` for new components in Mozilla Firefox. `Rust` is preferred due to its memory- and thread-safety, high performance, and expressiveness. Following this principle, we are also adhering to NSSMod-10. We note that the use of the `Cryptoki` API mandates the use of `C` FFI bindings, which generally require `unsafe` code blocks, as mentioned in Section 5.3.1 for OSSLProv-12. Our design will isolate these `unsafe` blocks into the smallest possible units and encapsulate them within convenient `Rust` functions and methods.

**Designated Development Platform.** Our design aligns with NSSMod-11 by compiling the QUBIP `Module` as a dynamically loadable module. We will focus on Linux, specifically Fedora Linux, as the primary development and testing platform for the QUBIP `Module`, in accordance with NSSMod-12. This focus is important because end-users of the *Quantum-secure Internet browsing* pilot demonstrator of WP2 will interact with Mozilla Firefox (and indirectly the QUBIP `Module`) in a Fedora Linux-based environment. Therefore, ensuring compatibility and smooth operation on Fedora Linux is a high priority. While support for other platforms, such as MacOS and MS Windows (provided they are supported by the selected back-end projects), is desirable, it is a non-requirement for the scope of the QUBIP `Module` design. However, although we cannot guarantee portability to other platforms, our design strives to follow best practices in terms of portability to facilitate future porting efforts.

**Interoperability Testing.** Finally, our interoperability testing involves evaluating our QUBIP `Module` design at the TLS 1.3 protocol level against the OpenSSL and the QUBIP `Provider` stack. Due to the design matching OSSLProv-18 of Section 5.3.1, our QUBIP `Module` will indirectly ensure ongoing interoperability with other selected PQ/T Hybrid stacks and protocol implementations, thereby fulfilling NSSMod-13.

### 5.3.4. Mbed-TLS: Design

Mbed-TLS is built with a highly modular architecture, which means that each component is designed as a separate module. This modularity allows developers to include only the components they need, reducing the memory footprint and increasing efficiency. The list of modules includes, among others, the *cryptographic algorithms module* and the *platform abstraction layer*. The former provides implementations of various cryptographic algorithms used in SSL/TLS and other security protocols, while the latter abstracts platform-specific features, enabling Mbed-TLS to be portable across different systems. As described hereafter, these modules will be used to address the requirements set in Table 5.4.

One of the strengths of Mbed-TLS is its configurability. Through a single configuration file (`config.h`), users can enable or disable features and modules. This configurability allows developers to tailor the library to specific needs, removing unnecessary features to save space, and optimizing the final build for performance or memory usage based on the target application. We will implement the necessary configuration options in the config.h file, in order to support the transition to PQC.

For Mbed-TLS, it has been decided to define a specific framework and design, and reduce the implementation only to the necessary building blocks. This is different from OpenSSL, which is intentionally designed to maximizing cryptographic agility to dynamically choose primitives and modules (see Section 5.3.1).

The cryptographic algorithms module handles today's classical cryptographic suites. We will modify this module with new APIs, which provides the PQC algorithm implementations. This design choice allows

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

seamless selection of classical and/or PQ algorithms from the Mbed-TLS configuration file (see requirement MbedTLS-01).

In order to make the integration of PQC as transparent as possible (see requirement MbedTLS-02), we will expose the cryptographic API primitives to higher level applications following the guidelines for Mbed-TLS interfaces and APIs that support extension and customization.

In addition, we will implement a new abstraction layer that will be used by the Mbed-TLS *alternate functions* to take full advantage of the cryptographic primitives implemented by the HW SE, so that it will be possible to easily select these primitives from the configuration files (see requirement MbedTLS-03).

The Mbed-TLS implementation will expose `X25519` and `EdDSA-25519` for classical cryptography, while it will expose ML-KEM and ML-DSA for PQC. The same functionalities will be provided by both versions, with or without SE.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

# 6. Operating System

## 6.1. Introduction

This chapter of the document differs from the others as it describes the existing OS workflow and requirements for the component integration. The OS workflow and the requirements are not a subject to change during the development. This chapter doesn't create any PQ-related requirements by itself.

OSs provide an embracing self-consistent environment for various components to work in an interoperable manner. At the start moment of the project, there were no OSs providing a full or near-full application stack usable for the PQ transition. To ensure the PQ transition, it is necessary to add new components and adjust the existing components to make them PQ-capable. Red Hat as a QUBIP partner has chosen **Fedora Linux** [77] as the OS that the PQ-capable components will be added to.

The OS provides mechanisms to install and configure the PQ-capable components, both included in the distribution itself and 3rd-party components. As a rule, these components are not developed by the OS developers but are chosen according to the distribution rules and maintained afterward. All components must be in a similar form as the other components of the OS for better user experience of PQ algorithms.

We address the lack of necessary software components and their integration to make PQ/T hybrid TLS v1.3 available in the OS. This means integrating the pluggable modules as described in Section 5.3 into the stack to make them available to user application to establish quantum-secure communication channels by either extending the plug-in mechanisms or hard-coding the necessary abilities to get the algorithms usable in the TLS. We intend to provide the necessary modifications to the corresponding upstream to ensure the solutions will be suitable for wide usage and reduce the maintenance burden. Until the full standardization process is finalized, the implementations both included in the distribution and 3rd-party have some experimental status.

## 6.2. Requirements

The scope of the work involves extending the capabilities of the crypto libraries of low-level components to provide PQ algorithms. The applications relying on these crypto libraries (**OpenSSL**, **NSS**) will be suitable to use the PQ algorithms via standard interfaces to the extent it is implemented by their maintainers, e.g., OpenSSL-based web-server `nginx` and OpenSSL-dependent command line TLS client tool `curl` accept the configuration options for specifying key exchange algorithms and can use OpenSSL providers basing on system-wide configurations (OpenSSL system-wide configuration and crypto-policies, see below).

Fedora Linux MUST have a capability to install and configure the components providing PQ and/or hybrid algorithms and use them from applications for establishing PQ-protected TLS connections.

There are several possible options for installing the necessary components:

1. *Parts of the distribution*. Components may be added to Fedora Linux according to the standard procedure, published as a part of the standard OS repository, and installed in a regular way.
2. *3rd-party repository maintained by partners*. These repositories may be added to a particular installation as a part of the process of system configuration and the components provided this way may be installed in a regular way after that.
3. *Flatpak format packages* [78]. They can also be available via dedicated repositories configured separately. This format may be preferable for Graphical User Interface (GUI) applications. Also,

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

applications distributed in such a format may have modified versions of system libraries available only for such applications.

The components MUST be compatible with the declared version of the Fedora Linux.

As Fedora Linux implements system-wide “crypto policies” providing consistent setup for all the applications using standard backends (OpenSSL and NSS), separate crypto policy/policies enabling PQ and/or hybrid algorithms MUST be implemented. Components provided as a part of distribution MUST be compatible with the crypto policy/policies mentioned before. Components provided via 3rd-party repositories SHOULD be compatible with the crypto policy/policies mentioned before (e.g., providing algorithm names matching the ones enumerated in policies). Components provided via Flatpak repositories MAY be compatible with the crypto policy/policies mentioned before. The components we consider the outcome of work SHOULD be interoperable. By using the term “interoperable” we mean that there MUST be a subset of PQ and/or hybrid algorithms that, being configured on both sides, give us a possibility to establish connection.

## 6.3. Design

We ship the following components as a publicly available part of Fedora Linux distribution:

- **liboqs** – a library providing a low-level implementation of the PQ/Hybrid algorithms. The Fedora Linux build of liboqs includes the algorithms standardized by NIST [79]. Other algorithms may be added to the component despite lack of formal approval, in case when they are widespread enough for compatibility testing.
- **oqsprovider** – an OpenSSL provider based on liboqs making OpenSSL and OpenSSL-based applications (`nginx`, `apache`, `curl`) suitable to work with the PQ algorithms.

**liboqs** was chosen to be included in the Fedora after investigation of the possibilities. It is written in C language, follows best development practices, provides a wide list of the algorithms, has a suitable license, and a very responsive upstream. The combination of these circumstances make liboqs the best one both for QUBIP purposes and for possible future use in Fedora. Also, liboqs uses the same low-level implementation of the PQ algorithms that is, according to the best of our knowledge, planned to be included into NSS.

**oqsprovider** is based on liboqs, it is implemented by the same team, and the tests using oqsprovider are run as a part of OpenSSL integration tests.

The level of matching the standards is the one provided by the version of liboqs available in the distribution. Usually the latest version of the liboqs and the oqsprovider is available in Fedora Rawhide [80] i.e., a development version.

Having these components as a part of the distribution makes Fedora Linux a good foundation for other PQC research projects.

We plan to make available the pluggable components described in Chapter 5 via a 3rd-party repository.

In case the version of NSS or Mozilla Firefox will result to be incompatible with the system-level one, we plan to distribute the PQ-capable build of Mozilla Firefox web browser in a Flatpak format.

To simplify the setup of PQ-capable installation, we plan to prepare container images that have all the necessary components in addition to crypto policies being configured to the fullest extent possible within feasibility. These images will have a relevant documentation describing the configurations steps that should be performed manually, if necessary, to finalize the configuration and run the relevant test scenarios.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

# 7. Firefox Browser

## 7.1. Introduction

Mozilla Firefox [81] is a widely-used, open-source web browser developed by the Mozilla Foundation. Initially released in 2002, Mozilla Firefox has grown to become one of the most popular browsers globally, renowned for its emphasis on privacy, security, and user-centric features. It supports a diverse range of web standards and technologies, providing users with a fast, secure, and customizable browsing experience. Mozilla Firefox's commitment to open-source principles allows it to benefit from a vibrant community of developers and contributors, continually enhancing its functionality and performance. Furthermore, Mozilla Firefox also integrates a robust extension ecosystem, and many other advanced features, making it a versatile choice for both casual and power users.

The cryptographic stack of Mozilla Firefox is based on NSS [61], but their current integration offers a lower degree of cryptographic agility, compared to the related OpenSSL `Provider` solution (described in Section 5.3.1). In QUBIP we plan to explore these limits, designing and implementing patches to overcome them, in coordination with the Mozilla Firefox maintainers, to ensure our changes meet the project requirements and can be contributed upstream.

## 7.2. Requirements

We collect the requirements for the PQ transition of Mozilla Firefox in Table 7.1. These requirements provide a framework for the planned implementation, which is detailed in Section 7.3.

**Table 7.1:** Firefox browser requirements

| Req. | Description |
|---|---|
| MZFF-01 | QUBIP's Mozilla Firefox build MUST be able to leverage the QUBIP `Module` (through the NSS layer) for PQ/T Hybrid KEMs in TLS 1.3 handshakes |
| MZFF-02 | QUBIP's Mozilla Firefox build MUST be able to leverage the QUBIP `Module` (through the NSS layer) for PQ/T Hybrid Signatures in TLS 1.3 handshakes |
| MZFF-03 | QUBIP's Mozilla Firefox build MUST be able to leverage the QUBIP `Module` (through the NSS layer) to inspect, evaluate, validate, and verify certificates for PQ/T Hybrid PKI |
| MZFF-04 | QUBIP's Mozilla Firefox build SHOULD be able to leverage NSS to dynamically discover the algorithms supported by the loaded QUBIP `Module`, to expose them for use at the protocol and application layers |

## 7.3. Design

QUBIP's Mozilla Firefox build will facilitate the transition to PQC through NSS. From a high-level perspective, we aim to modify the source code of Mozilla Firefox as little as possible. Instead of making extensive modifications, we are designing loadable modules for NSS that leverage its native support for PKCS #11. The PKCS #11 standard defines an API known as `Cryptoki` [67], which provides a standard interface for

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

managing and using cryptographic tokens. In our design, the module acts as a software token, utilizing an external cryptographic library to provide PQC capabilities and exposing our implementation of the `Cryptoki` API for NSS to use. Our design aims to create a framework for integrating external cryptographic libraries, making it easy for developers to test different PQC implementations.

Additionally, using loadable modules offers several advantages, such as simplicity in design, ease of maintenance, and greater flexibility for future updates and enhancements. For a more in-depth perspective on NSS refer to Section 5.3.2, and for the loadable modules, refer to Section 5.3.3.

Our design ensures that QUBIP's Mozilla Firefox build can leverage the QUBIP `Module` through the NSS layer to achieve the requirements in the Table 7.1.

For MZFF-01 and MZFF-02, the design integrates PQ/T Hybrid KEM and PQ/T Hybrid Signatures into TLS 1.3 handshakes by utilizing the QUBIP `Module`'s capabilities via NSS.

For MZFF-03, our build will be able to inspect, evaluate, validate, and verify certificates for PQ/T Hybrid PKI, ensuring robust and secure certificate management. We will ensure that our build will be able to handle such certificates programmatically, but changes for comprehensive GUI support for PKI facing end-users are not planned within the scope of QUBIP. The above limitation is necessary as we anticipate that additional requirements regarding the user experience for interacting with hybrid PKI may arise very late during the QUBIP timeline, as discussions around PQ/T Hybrid certificates evolve.

Lastly, for MZFF-04, our design leverages NSS to dynamically discover the algorithms supported by the loaded QUBIP `Module`, exposing them for use at the protocol and application layers, thus providing flexibility and extensibility for future cryptographic advancements.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

# 8. Self-Sovereign Identity

## 8.1. Introduction

The Self-Sovereign Identity (SSI) reference model [82] consists of three layers, depicted in Figure 8.1. Each layer contributes to the generation of the identity and defines the basic principles for trustworthy interactions between peers.

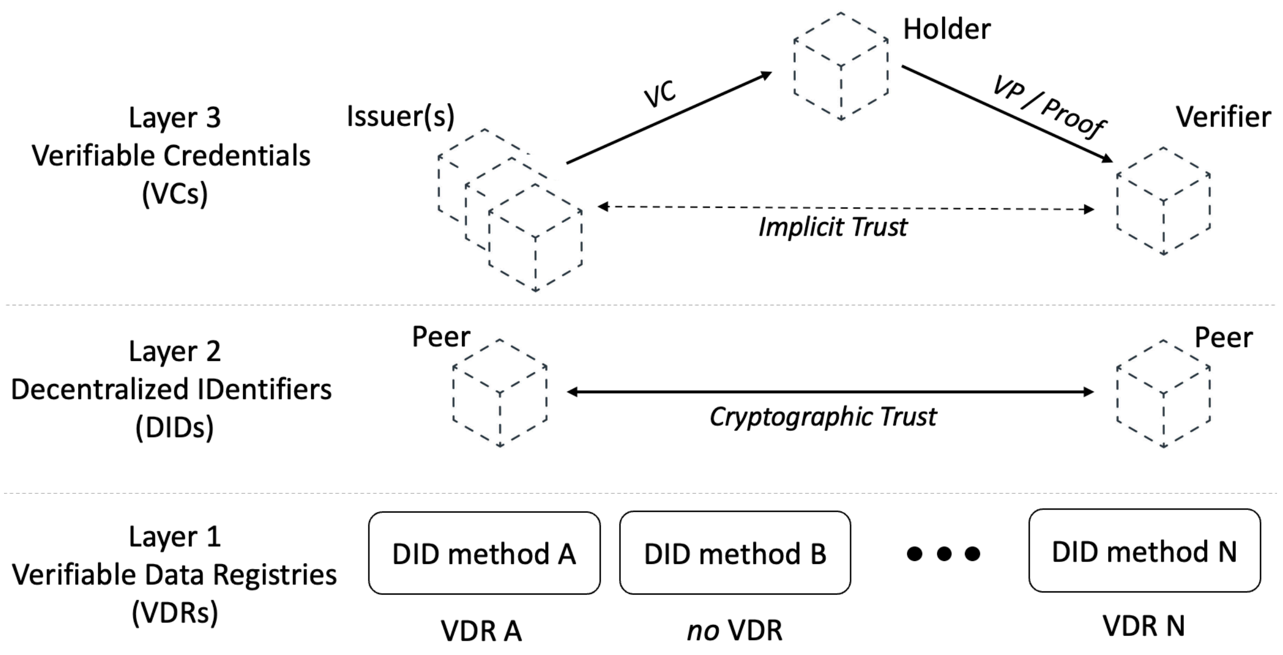


**Figure 8.1:** The Self-Sovereign Identity reference model.

Layer 1 is implemented by means of any Verifiable Data Registry (VDR) that is used to store the public identity data. The Decentralized IDentifier (DID) [83] is a new type of globally unique identifier designed to verify a peer. The DID is a Uniform Resource Identifier (URI) in the form

```
did:method_name:method_specific_id
```

where `method-name` is the name of the DID method used to interact with the VDR and `method-specific-id` is the pointer to the DID document stored in the VDR. Thus, a DID associates a peer with a DID document [83] to enable trusted interactions with it.

Appendix A shows an example of a DID document containing the DID and two verification methods (i.e., public keys) for authentication purpose.

The DID method [83, 84] is a software implementation used by a peer to interact with the VDR of choice. Following World Wide Web Consortium (W3C) recommendation [83], a DID method provides the so-called CRUD functionalities to

- **Create** a DID: generates an identity key pair $(sk, pk)$, the corresponding DID document containing the public key $pk$, and stores the DID document in the VDR at the `method-specific-id` pointed to by the DID,

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

- **Resolve** a DID: retrieves the DID document from the `method-specific-id` on the VDR pointed to by the DID,
- **Update** a DID: updates the content of the DID document, e.g., generates a new key pair $(sk', pk')$, and stores the modified DID document to the same or a new `method-specific-id` if the node needs to change the DID, and
- **Deactivate** a DID: provides an evidence in the VDR that the DID has been revoked by the owner.

The DID method implementation is VDR-specific and makes the upper layers independent of the VDR of choice. The DID methods [85, 86] using VDRs implemented by a Distributed Ledger Technology (DLT) leverage the immutability features of the DLTs that acts as the RoT for DID documents. However, the VDR can also be implemented by a web domain. The `did:web` method [87] leverages a web domain for the RoT of DID documents; with this option, it is up to the implementer to secure the integrity of DID documents according to best practices. Another option exists with `did:key` method [88], in this case the `method-specific-id` components of the DID coincides with the encoded public key, hence the VDR is not required.

As shown in Figure 8.1, Layer 2 uses DIDs and DID documents to establish a cryptographic trust between two peers. In principle, both peers prove the ownership of their private key $sk$ bound to the public key $pk$ in their DID document.

While Layer 2 uses DIDs to start authentication, Layer 3 completes it and also deals with authorization to services/resources using Verifiable Credentials (VCs) [89]. A VC is an unforgeable digital credential that contains additional characteristics of the digital identity of a peer than its key pair $(sk, pk)$ and DID. A VC contains the metadata to describe properties of the credential (e.g., context, id, type, issuer, issuance, and expiration dates), most importantly, the DID and the claim(s) about the identity of the peer in the `credentialSubject` field, and the information for a Verifier of the VC to check the status of revocation in the `credentialStatus` field. The addition of a proof, with the digital signature made the Issuer of the VC, makes it tamper-evident and trustworthy. Appendix A shows an example of VC.

The combination of identity key pairs $(sk, pk)$, DID, and VCs shapes the digital identity of a peer in the SSI ecosystem.

Layer 3 works in accordance with the Triangle-of-Trust (ToT) depicted in Figure. 8.1. Three different roles coexist:

- **Issuer(s)** asserts claims about a peer, creates a VC from these claims, and issues the VC to the Holder. In addition, the Issuer manages VC revocation [89] and provides evidences for Verifiers to check the revocation status [90] of all issued VCs without compromising the privacy of the Holders.
- **Holder** owns one or more VCs and generates a Verifiable Presentation (VP) [89] to authenticate with a Verifier and request services/resources. A VP is constructed as an envelope of the VC issued by an Issuer, where the proof contains the Holder's signature. Appendix A shows an example of VP.
- **Verifier** receives a VP from the Holder and verifies the authenticity by checking the signature made by the Issuer on the VC and by the Holder on the VP, checks the revocation status of the VC and the claim(s) and metadata before granting or rejecting access to the Holder. The Verifiers have an implicit trust on the Issuer(s).

It must be noted that a credential system based on *plaintext* VCs does not protect the privacy of the Holder because the VC, the contained claims, and the signature of the Issuer are exchanged in plain text. In this sense, a *plaintext* VCs result in linkability and traceability of peers. The *anonymous* VCs represent a privacy-preserving alternative, that enables the Holder to manage its VC by choosing the level of information disclosure. In detail, the roles in the Triangle-of-Trust are modified as follows:

- **Issuer(s)** asserts the capability of a Holder to prove its identity, possibly based on the knowledge of some secret attributes. After proper verification, the Issuer issues the *anonymous* VC to the Holder, and manages the VC revocation without compromising the privacy of the Holder.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

- **Holder** owns one or more *anonymous* VCs and can take advantage of Zero-Knowledge (ZK) primitives and protocols to interact with Issuer(s) and Verifier(s) in a privacy-preserving way. This means that the Holder can obtain a VC and prove that its identity satisfies certain properties, still maintaining the desired level of privacy. In fact, these operations can happen in a completely anonymous way (i.e., without revealing any identity details) or by *selectively disclosing* only a subset of the claims contained in the VC (i.e., without revealing some specific secret attributes). Moreover, the Holder can prove to the Verifier that its VC is legit, that means it contains a valid signature generated by an Issuer using a Proof of Knowledge of a Signature (PoKS) [91].
- **Verifier** receives a PoKS from the Holder and verifies that the Holder actually holds a valid VC, containing the undisclosed secrets and a valid signature from an Issuer.

In general, the SSI reference framework involves peer-to-peer interactions, but it is also intended to be used in the client-server architecture typical of the Web. In this scenario, the SSI is a promising decentralized alternative for implementing client authentication. The client establishes a secure communication channel with the server using TLS [46] with server authentication only. Then, assuming the client already has its self-sovereign identity, it creates a VP, signs, and sends it to the server for authentication. Upon successful authentication, the server will also check the client's VC claim(s) for authorization before granting access to the requested service/resource.

QUBIP takes a practical step towards the transition of the SSI ecosystem by designing and implementing transition to PQC of *plaintext* VCs with a solely PQ and PQ/T hybrid approaches, and of PQ *anonymous* VCs with selective disclosure capabilities. The transition exercise will target the IOTA Identity library [92]. This is a widely used SSI library written in the `Rust` programming language and is the result of a large open source, community-led SSI project maintained by the IOTA Foundation. The library provides all the functionalities to handle W3C compliant DIDs, DID documents, VCs, and VPs. In essence, it is a general purpose SSI library and, therefore, the perfect target for a practical transition to PQC of the SSI ecosystem.

## 8.2. Requirements

### 8.2.1. Plaintext Verifiable Credentials

CRQCs threaten traditional cryptography mainly through the Shor's algorithm [93], for factoring integers and solving discrete logarithms, and Grover's search [41], which can help speed-up unstructured search problems. The main cryptographic technologies in SSI with *plaintext* VCs are traditional asymmetric signatures to authenticate identity. As a result, SSI is vulnerable to Shor's algorithm, to stripping attacks in case of a PQ/T hybrid VC, but not to the store-now-decrypt-later attacks.

Therefore, the transition to PQC of the SSI ecosystem, with *plaintext* VCs, means switching from traditional to PQ or PQ/T hybrid key pair generation, signature generation, and verification. Switching means select and adopt appropriate PQ signature algorithms to be used by the Issuer and Holder to sign the VC and VP respectively, and by the Verifier to verify these signatures for authentication purposes.

In the target Web scenario, a Holder will request access to a Verifier with a frequency that is higher than the frequency with which they request the issuance of a *plaintext* VC from the Issuer. We considered this assumption to define the requirements for the transition.

The list of requirements is reported in Table 8.1.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

**Table 8.1:** Requirements for SSI with plaintext VC

| Req. | Description |
|---|---|
| SSI-PLAIN-01 | The signature verification time MUST be as short as possible to increase the number of Holders a Verifier can authenticate in a unit of time |
| SSI-PLAIN-02 | The signature generation time MUST be as short as possible to (*i*) increase the number of VCs an Issuer can issue in a unit of time, and to (*ii*) decrease the time a Holder spends preparing the VP |
| SSI-PLAIN-03 | The signature size SHOULD be as small as possible to reduce the overall size of the VC and VP |
| SSI-PLAIN-04 | The public key size SHOULD be as small as possible to reduce the size of the DID document |
| SSI-PLAIN-05 | Transition MUST provide a solution solely based on PQC, and OPTIONAL a PQ/T hybrid alternative |

### 8.2.2. Anonymous Verifiable Credentials

*Anonymous* credentials schemes based on traditional cryptography, such as CL [91] or BBS$^+$ [94], are not suitable in the light of the development of a CRQC.

Moreover, while for *plaintext* VCs the transition to PQC amounts to a proper migration from traditional to PQ or PQ/T hybrid key-pair generation, signature generation, and verification (see Section 8.2.1), for *anonymous* VCs, the situation is much more involved.

To understand this, it is crucial to recognize that *anonymous* VCs require proving knowledge of a signature on some (potentially secret) attributes. In an anonymous credential system, the user receives a signature on their attributes and a secret key at issuance. To show their credentials, they reveal the requested attributes and prove knowledge of the signature, hidden attributes, and secret key to remain anonymous.

At the core of this mechanism is a PoKS, which consists of a “signature scheme with efficient protocols” (informally, a digital signature scheme with specific features such as the ability to sign committed hidden messages and to prove knowledge of a signature on such messages) and an associated ZK proof system, as described (in the classic setting) by Camenisch and Lysyanskaya [94].

While there are many practical proposals for PQ ZK proof systems, the state of PQ “signature schemes with efficient protocols” is still at the frontier of research. First, anonymous credential schemes based on PQC have been proposed only recently (not before 2022) [95, 96, 97, 98], with the exception of [99], which is however impractical due to having a signature size of at least 670 MB [95, p. 4]. Second, the literature on PQ anonymous credential schemes is quite scarce, essentially only the aforementioned papers. Third, none of the papers introducing these schemes is accompanied by a software implementation, nor does any of them take into account implementation issues such as side-channel attacks.

For these reasons, in order to identify a suitable candidate scheme for PQ *anonymous* VCs and carry out a software implementation, which will be the first of its kind, we focused mainly on three aspects (*i*) the cryptographic assumptions on which the schemes base their security, (*ii*) the signature size, and (*iii*) the completeness and clarity of the pseudocode presented in the papers proposing the schemes.

Regarding the cryptographic assumptions, we focused our attention on those used by the schemes selected for standardization by the NIST at the end of the third round. In particular, they have chosen four algorithms: one KEM, KYBER [100], whose security is based on lattices, and three digital signatures, Dilithium [101], Falcon [13] and SPHINCS$^+$ [35], whose assumptions are based again on lattices and cryptographic hash functions. We examined different articles from the recent literature and we found three valid candidates [95, 96, 97], all based on lattice assumptions. Here, we report their main characteristics:

1. In the paper [95], the authors present a lattice-based framework for anonymous credentials, improving the previous post-quantum state-of-the-art [99]. The size of a proof is around 640 KB.
2. In the paper [96], the authors improve the previous work [95] with the aid of a new cryptographic primitive, called Commit-Transferable Signatures (CTS) to realize this paradigm more efficiently. They obtain a proof with dimension around 500 KB.
3. The papers [97, 98] from Bootle, Lyubashevsky, Nguyen, and Sorniotti (BLNS) take a different approach from the previous two proposals, and their scheme exhibits shorter credentials, with a size of around 125 KB. The security of the scheme is based on a new assumption, which is a variation of classic lattice problems. From now on, we will refer to this proposal as the BLNS framework, for conciseness.

The three proposals share important ideas in common, for example:

- all three schemes use a Non-Interactive Zero-Knowledge proof (NIZK) derived from the NIZK proposed in [102];
- all three scheme utilize the Ajtai commitment, introduced in [103];
- to realize a framework for anonymous credentials, both [95] and [96] exploit the '*signature with efficient protocol paradigm*' introduced in the seminal work of Camenisch and Lysyanskaya [91];
- both [96] and [97] use the same sampling trapdoor algorithm described in [104]; in particular, the sampler is taken from [105].

On the other way around, one of the biggest distinctions between [97] and [95, 96] concerns the cryptographic assumptions. While all three of them are based on the Learning with Errors (LWE) and the Short Integer Solution (SIS) assumptions, perhaps the two most studied lattice assumptions, the BLNS framework [97] also uses the newly introduced family of assumptions Inhomogeneous SIS$_f$ (ISIS$_f$), parametrized by a function $f$, that allow to demonstrate significantly fewer and simpler relations.
The assumption on which the BLNS framework [97] relies on, ISIS$_f$, is a very natural generalization of the underlying problem upon classic lattice-based signature schemes such as [104], and it is also similar to other recently, and independently, proposed lattice assumptions [106]. Furthermore, under specific choices of $f$, the new assumption is proven to be equivalent to SIS, a well-known assumption introduced by Ajtai in 1996 [103], on which many constructions base their security, like the standardized signature Dilithium [101] relying on the variant $\mathsf{SelfTargetMSIS}$.

### 8.2.3. Revocation Mechanism for Anonymous Verifiable Credentials

One of the most important functionalities that an *anonymous* credential system must possess is the revocation. An efficient revocation mechanism represents a crucial requirement in any credential system, independently of any privacy-enhancing features it offers (e.g., anonymity or selective disclosure). The possible reasons why a credential needs to be revoked can be grouped in three categories:

1. *Natural expiration* – it is the most common case, where the credential reaches its expiration date, thus becoming outdated and unusable.
2. *Issuer-initiated revocation* – the Issuer revokes the credential before its expiration date, because for some reason the Holder has lost its permission to use the credential (e.g., a driving license revoked because of speeding).
3. *Holder-initiated revocation* – the Holder asks for a revocation of its credential, for instance in the case of credential theft or because its secret key has been compromised.

While the W3C standard approach to revocation [90] can also be adopted with PQC-based *plaintext* VCs, essentially because it does not rely on cryptography vulnerable to CRQC, the same approach is not suitable for privacy-sensitive contexts.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

Two main approaches suitable to implement an efficient revocation mechanism for *anonymous* VCs exist:

1. **Accumulator approach**: Camenisch and Lysyanskaya in [107] propose the use of a *dynamic accumulator* to solve the revocation problem. The cost of an update for a dynamic accumulator, and for the related witnesses, is linear in the number of users added to or revoked from the system; the update of the witness must be done by the Holder. The latter is a defect, especially if the revocation operation is common and involves many Holders. In [108], Camenisch, Kohlweiss and Soriente modify the dynamic accumulator introduced in [107], such that there is almost no cost for the Holder or the Verifier, and the computation done by the Issuer is limited. In this case, the update operation does not require the use of any secret key, hence this operation may be performed by any untrusted entity. Both the solutions in [107] and [108] are based on pre-quantum assumptions.
2. **Timestamp approach:** Camenisch, Kohlweiss and Soriente in [109] had yet another idea to implement a revocation mechanism, based on timestamps. Each credential is valid only for a time interval called *epoch*, and a new credential must be issued for every epoch. The revocation consists in non-issuing the credential for the next epoch. Even if this approach seems less efficient, in [109] a non-interactive solution is proposed, where the Issuer publishes an *updating material* that the Holders can retrieve to update their credential with no additional interaction with the Issuer. In this case, the credential contains a signature of the Issuer on a number of attributes, some of them chosen by the Issuer. Credential revocation is implemented by encoding a validity time property (i.e., a timestamp) into one of the Issuer-controlled attributes. Thus, an Issuer can periodically update valid credentials off-line and publish a small per-credential update value, for instance on a public bulletin-board. For the Issuer, this process consists in:
   a) the update of the validity time property for each non-revoked credential;
   b) the re-computation of a small portion of the Issuer signature on the updated credential;
   c) the publication of such small signature update in a set of updates for all the non-revoked credentials, identified with unique serial numbers or, preferably, by means of pseudonyms.

   A Holder can later download the update and refresh its credential to prove possession of a valid credential for the current time period.

## 8.3. Design

### 8.3.1. Plaintext Verifiable Credentials

#### 8.3.1.1. PQ approach

Table B.1 in Appendix B shows various statistics that we have experimentally measured on `liboqs-rust`, a `Rust` wrapper for the `liboqs` C library [110], to make the proper selection of the PQ signature algorithms.

In view of the postponement of FN-DSA (i.e., FALCON [13]) standardization announced by NIST, we have decided to exclude it from the selection at this early stage of QUBIP in order to prioritize the PQ signature algorithms, which are closer to final standardization, thus ensuring both current security requirements and timely implementation of the project.

The combined analysis of the requirements[1] presented in Table 8.1, and of the experimental evaluation of NIST selected PQ signature algorithms, presented in Table B.1, suggests the use of ML-DSA [44], in particular, ML-DSA-44 for NIST security level 2, the use of ML-DSA-65 for level 3, and the use of ML-DSA-87 for level 5.

[1]We analyzed the requirements in order of priority from SSI-PLAIN-01 (higher priority) to SSI-PLAIN-04 (lower priority).

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

#### 8.3.1.2. PQ/T hybrid approach

Although the PQ algorithms selected by the NIST have undergone rigorous reviews in recent years, they are not as much mature as the traditional algorithms. This fact has led us to consider hybrid schemes that combine both PQ and traditional signature algorithms in a single cryptographic scheme [30]. The underlying idea is that the hybrid scheme is secure as long as the security of one of the algorithms holds.

In SSI the goal of hybrid signature schemes is hybrid authentication, which is the property that authentication is achieved by the hybrid signature scheme provided that at least one signature algorithm remains secure. In other words, an adversary must violate both schemes to forge a credential and impersonate another user's identity. This way, if the PQ cryptographic assumption is found to be flawed in the future, the composition of the signature algorithms ensures that the scheme is as secure as the traditional signature algorithm.

There are several ways to combine algorithms to build a hybrid scheme [111]. Here, we design a hybrid scheme that combines PQ and traditional signatures using the concatenation combiner and achieving the Weak Non-Separability (WNS) property, customizing the design principles addressed in [32] to the SSI model. The Non-Separability property defined for hybrid signatures in [112] prevents an adversary from removing the signature generated by the secure algorithms, forcing the Verifier to rely only on the signature generated by the broken algorithm. This is commonly referred to as stripping attack. Among the different flavors, the WNS property implies that an adversary cannot remove one of the signatures without the Verifier noticing.

WNS is achieved through the adoption of an artifact. It is the evidence of the will of Issuers and Holders to hybridize their signatures on VC and VP, respectively. Given the SSI working principles introduced in Section 8.1, our design places the artifact at the protocol level and, specifically, within the DID document.

Here, we define the `CompositeSignaturePublicKey` type of verification method, a novel type for hybrid authentication purpose. It is designed to store a `compositePublicKey` object containing (*i*) the PQ and traditional public keys JSON Web Key (JWK) encoded, and (*ii*) the `algID` string representing the name of the algorithms used to generate the hybrid signature.

```
"id": "did:method_name:method_specific_id",
"authentication": [{
  "id": "did:method_name:method_spec_id#keys-1",
  "controller": "did:method_name:method_spec_id",
  // the new type in the verification method
  "type": "CompositeSignaturePublicKey",
  "compositePublicKey": {
    "algID": "id-MLDSA44-Ed25519-SHA512",
    "pqPublicKey": {
      // contain a JsonWebKey
      "kty": "ML-DSA",
      "alg": "ML-DSA-44",
      "kid": ".. key thumbprint ..",
      "pub": ".. encoded public key .."
    },
    "traditionalPublicKey": {
      // contain a JsonWebKey
      "crv": "Ed25519",
      "x": ".. x coordinate ..",
      "kty": "OKP",
      "kid": ".. key thumbprint .."
    }
}]
```

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

The `algID` in the above example of DID document represents the composition of ML-DSA-44 and Ed25519 signature algorithms, see Table 3 in [32] for details on the string values.

Let us define $m$ the message to be signed as the serialization of the VC or VP, $A_{pq}$ the PQ signature algorithm, $A_t$ the traditional signature algorithm, and $(sk_{pq}, pk_{pq})$ and $(sk_t, pk_t)$ the PQ and traditional key pairs, respectively. The PQ/T hybrid signature $\sigma_h = (\sigma_{pq}, \sigma_t)$ is the concatenation of the PQ signature $\sigma_{pq}$ and the traditional signature $\sigma_t$, calculated as it follows:

$$\begin{aligned} m' &= \text{Hash}(m) \\ \sigma_{pq} &\leftarrow \text{Sign}(sk_{pq}, A_{pq}, \text{DER}(\texttt{OID})||m') \\ \sigma_t &\leftarrow \text{Sign}(sk_t, A_t, \text{DER}(\texttt{OID})||m') \end{aligned}$$

where the Object IDentifier (`OID`) associated to `AlgID` is DER encoded, see Table 1 in [32].

Considering a stripping attack scenario where an adversary has compromised one of the two signature algorithms enough to forge the corresponding signature on the credential, the countermeasure relies on the Verifier only trusting the hybrid public key $(pk_{pq}, pk_t)$ and not the individual components. A Verifier resolving a DID to a DID document with a `compositePublicKey` must check the whole hybrid signature as it follows:

$$\begin{aligned} m' &= \text{Hash}(m) \\ \text{Verify}(pk_{pq},\ \text{DER}(\texttt{OID})||m',\ \sigma_{pq}, A_{pq}) \\ \text{Verify}(pk_t,\ \text{DER}(\texttt{OID})||m',\ \sigma_t, A_t) \end{aligned}$$

where the `OID` is selected based on the `algID` retrieved from the DID document.

The requirements and principles for the selection of the PQ signature algorithms previously discussed also apply in the case of a PQ/T hybrid approach to the transition. In addition, the authors of [32] provide guidelines to compose signature algorithms based on their bit-level security and to select the appropriate pre-hash to be applied to the message $m$.

All this information together suggests the adoption of two hybrid schemes using of ML-DSA and Ed25519: in particular, the schemes with `algID` equal to id-MLDSA44-Ed25519-SHA512 and id-MLDSA65-Ed25519-SHA512. The former composes ML-DSA-44 with Ed25519, while the latter composes ML-DSA-65 again with Ed25519. In both cases, the authors of [32] suggests using SHA512 as the pre-hash for consistency with Ed25519's internal use of this digest algorithm.

### 8.3.2. Anonymous Verifiable Credentials

In order to select the best framework for our needs, we have identified and carefully examined three papers [95, 96, 97]. In light of the considerations previously provided in Section 8.2.2, we chose the BLNS framework [97] as the best option. Indeed, despite the three analyzed proposals share many aspects in common, the BLNS framework excels in: having the shortest proof dimension (resulting in the smallest credential size, i.e., around 125 KB), having a higher degree of maturity in terms of implementability by providing several pages of pseudocode, and presenting an application suitable for the Web scenario.

From a cryptographic point of view, the BLNS framework has a common component with the PQ KEM N-th degree Truncated polynomial Ring Units (NTRU) [113, 114]. Such components are also implemented in the digital signature FALCON [13], which has been selected by NIST to be one of the next post-quantum standards. In particular, the BLNS framework [97] uses two functions from NTRU, called NTRU.TrapGen and GSampler.

NTRU is a lattice-based encryption scheme first proposed in 1996 [113, 114] and it has been presented to the NIST's call for post-quantum algorithms, reaching the third round as a finalist. Even if it has not

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

been chosen for standardization, NIST is very confident on its security guarantees, that are comparable to the next post-quantum standard KYBER: *"One of the difficult choices NIST faced was deciding between KYBER, NTRU, and Saber. All three were selected as finalists and were very comparable to each other. NIST is confident in the security that each provides. Most applications would be able to use any of them without significant performance penalties"* [115, page 18]. Moreover, NTRU was the first alternative candidate in case of patent issues of KYBER: *"If the agreements are not executed by the end of 2022, NIST may consider selecting NTRU instead of KYBER"* [115, Page 18, footnote 6]. This is a huge hint that, even if it has not been standardized, there are no arguments against its security, and the community agrees that the confidence level is high enough to use it in practical applications.

Furthermore, the BLNS framework is based on many different cryptographic assumptions that, under some additional hypothesis, are considered equivalent to the Module Short Integer Solution (MSIS) or the Module Learning with Errors (MLWE) assumptions [116]. Both MSIS and MLWE are also directly used by the BLNS framework to prove some security properties. These two are standard cryptographic assumptions, and their non-module versions (SIS and LWE) were introduced by Ajtai in 1996 [103] and Regev in 2005 [117], respectively. Their module version was instead introduced by Langlois and Stehlé in 2015 [116]. The BLNS framework uses these two assumptions various times to guarantee that the underlying protocols are secure. Since these assumptions are both used in the newly standardized algorithms ML-KEM [118] and ML-DSA [44], we are confident in their security and, nowadays, we could consider them "standard lattice assumptions".

Under the above assumptions, the BLNS framework is proven secure because it possesses the following security features.

1. *Anonymity.* Suppose that the Issuer and Verifier are malicious and possibly coallied. Moreover, suppose that there are two lists of the same length of attributes $\mathtt{attrs}_0$ and $\mathtt{attrs}_1$ and a subset of indexes $\mathtt{idx}$ such that for every index in $\mathtt{idx}$ the attributes of $\mathtt{attrs}_0$ and $\mathtt{attrs}_1$ are the same. The anonymity property says that the issuing and the verification of a set of attributes gives no information on the non-disclosed attributes of the Holder.
2. *One-More Unforgeability.* Any group with one or more malicious Holders, that presented multiple times some attributes of their VCs, is not able to produce a VP for a set of attributes that has not been issued by the Issuer. This means that the only way to produce a correct VP is from an honestly and correctly issued VC.

The above properties imply that the scheme is secure against any coalition including malicious Issuer(s), malicious Holder(s) and Verifier(s) who try to obtain some information about a target Holder. Moreover, any coalition of malicious Verifiers cannot link the same Holder across different verifications.

Finally, Figure 8.2 illustrates the high-level architecture and the dependencies of the functions involved in the BLNS framework. BLNS will be implemented for the demonstration of PQ *anonymous* VCs in the Web scenario.

Further details of the BLNS framework, its protocols, the relevant data, and the various functions considered in the implementation are provided in Appendix C.

### 8.3.3. Revocation Mechanism for Anonymous Verifiable Credentials

In order to implement an efficient revocation mechanism for PQ *anonymous* VC, there are two possible alternatives, as discussed in Section 8.2.3.

Since there are no PQ versions of the accumulator-based solution, we selected a timestamp approach similar to the one described in [109] that is PQ secure by construction (i.e., it does not rely on asymmetric cryptography). The timestamp approach can be adapted and implemented for the BLNS framework.

This solution has the advantage that the Verifier only checks that the VC is referred to the correct epoch, without having to check a revocation list. The cost of updating the VC is minimal for the Holder and

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

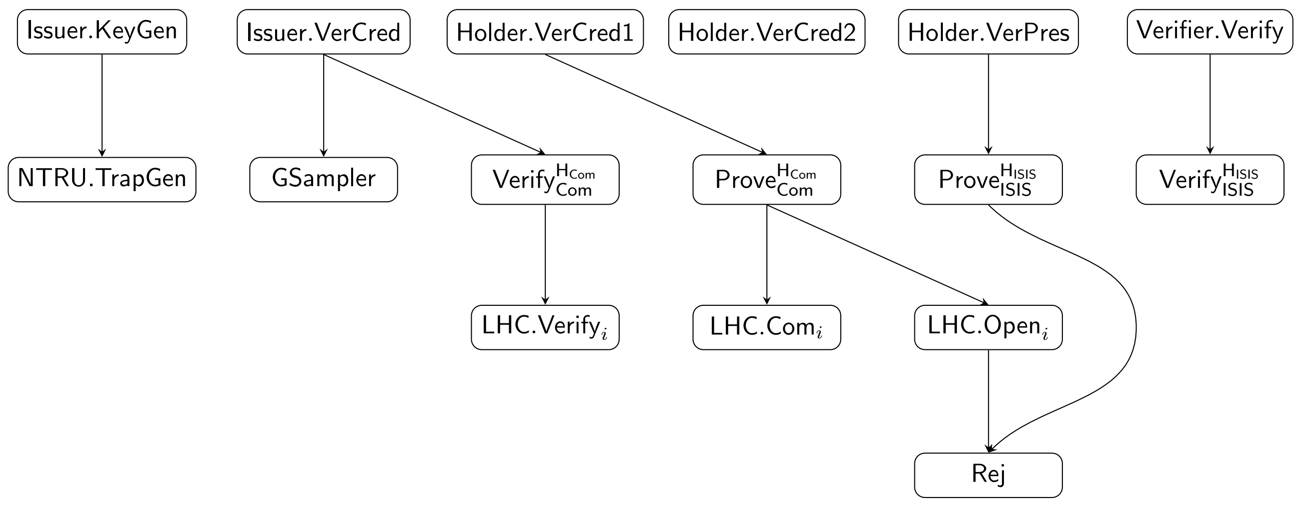

Issuer.KeyGen
Issuer.VerCred
Holder.VerCred1
Holder.VerCred2
Holder.VerPres
Verifier.Verify
NTRU.TrapGen
GSampler
$\mathsf{Verify}^{\mathsf{H}_{\mathsf{Com}}}_{\mathsf{Com}}$
$\mathsf{Prove}^{\mathsf{H}_{\mathsf{Com}}}_{\mathsf{Com}}$
$\mathsf{Prove}^{\mathsf{H}_{\mathsf{ISIS}}}_{\mathsf{ISIS}}$
$\mathsf{Verify}^{\mathsf{H}_{\mathsf{ISIS}}}_{\mathsf{ISIS}}$
$\mathsf{LHC.Verify}_i$
$\mathsf{LHC.Com}_i$
$\mathsf{LHC.Open}_i$
Rej

**Figure 8.2:** BLNS high-level architecture, involved functions and dependencies

are comparable to other solutions for the Issuer. Furthermore, this approach enables a *rich revocation semantic*, since the credential can be partially revoked and/or updated (i.e., only some VC attributes).

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

# 9. IKE-less IPsec

## 9.1. Introduction

Internet Protocol (IP) in both versions, IPv4 and IPv6, is the main protocol that allows communication as we understand today in Internet. The IP design lacks any security mechanism, and this is why IPsec [119] was introduced. IPsec protocol works at the network layer, protecting and authenticating IP packets. As a simple definition, IPsec is a framework of open standards, algorithm-independent, to provide data confidentiality, data integrity, anti-replay protection, and origin authentication. IPsec provides these features adding a new header directly in the IP packet (transport or host-to-host mode), or as part of an encapsulation of the original IP packet in a new one (tunnel mode or VPN). In both cases, the header carries the information that allows authentication and integrity verification of the IP packet payload, applying different algorithms [120]. Optionally IPsec can encrypt the payload. All protocols used in these processes belong to the symmetric cryptography family (confidentiality) or to hashing-related algorithms (integrity and authentication) and therefore are less affected by the development of a CRQC and Shor's algorithm [93]. The use of these algorithms does not require to migrate to quantum-secure alternatives, at this stage [121].

Internet Key Exchange (IKE) is an optional and complementary protocol associated with IPsec to implement key management strategies, see Figure 9.1. IKE main functions are session negotiation and establishment, peer authentication, and key generation and exchange. IKE allows user to define security policies in terms of algorithms to use, length of keys, and automate re-keying and deletion. The latest version IKEv2 [122] depends strongly on key exchange and asymmetric cryptography [123], requiring a PQ migration strategy.

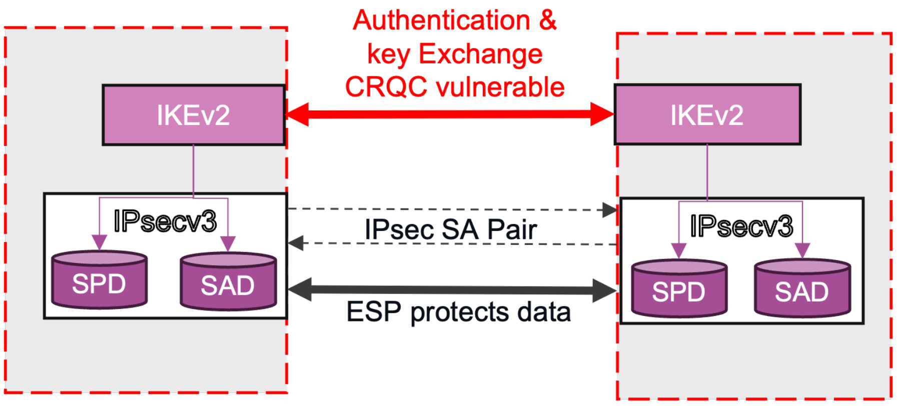


**Figure 9.1:** IPsec and IKE architecture.

QUBIP proposes a straightforward approximation to the transition of IPsec to PQC, by removing the IKEv2 protocol and substituting it with a Software-Defined Networking (SDN) solution based on standards (see Section 9.1.1), and with the support of the additional tools to apply quantum-secure mechanisms.

### 9.1.1. Centrally Controlled IPSec

A recent IETF standard [1], proposes an alternative that avoids the use of IKEv2 protocol (IPsec IKE-less), in an approach that uses the SDN concept. QUBIP proposes the adoption of this standardized approach as the starting point for an agile and transitional cryptography model.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

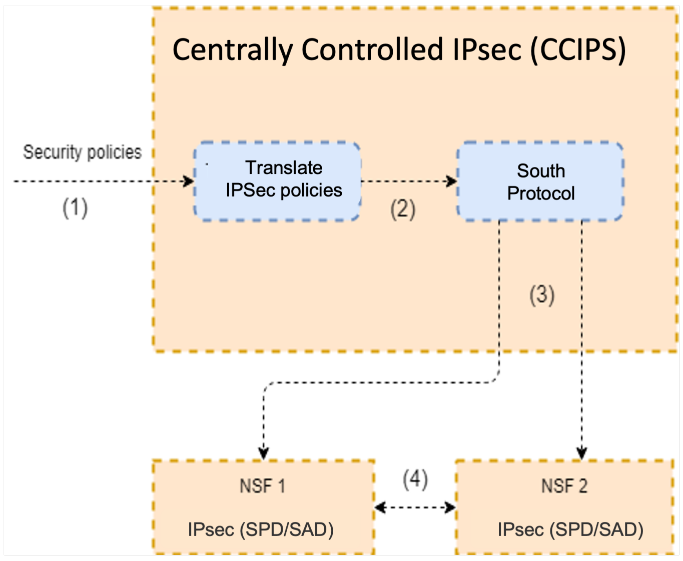


**Figure 9.2:** IPsec IKE-less model from RFC-9061 [1].

A simplified version of the IKE-less approach is represented in Figure 9.2. An administrator defines general flow-based security policies (1). These policies are translated by a centralized controller (e.g., SDN controller) to the specific IPsec entries for Security Policy Database (SPD) and Security Association Database (SAD), which contains the cryptographic algorithms type and key length for IPsec, the re-keying rules, and routing rules. Those configurations are delivered (2) to the component that will translate them to the Interface to Network Security Function (I2NSF) standardized format. Later, the controller sends them (3) to the IPsec endpoints Network Security Function (NSF) or Centrally Controlled IPSec (CCIPS) agents. Finally, the IPsec connection is established (4).

### 9.1.2. Hybridization

The hybridization of key material refers to the process of combining different cryptographic keys or key generation methods from multiple cryptographic systems to enhance overall security.

This concept is based on the principle that using keys generated or secured by diverse methods can provide stronger protection against potential attacks, including those that might exploit weaknesses in a single cryptographic system.

From the classical cryptography point of view, the key aspects include the combination of symmetric and asymmetric encryption. Asymmetric encryption uses a pair of keys (public and private keys) for encryption and decryption, and its security is based on a computationally hard mathematical problem, mainly factoring large numbers, but it presents the problem that it is slower for large data transfers. As a counterpart, symmetric encryption uses a single key for both encryption and decryption. It is much faster than asymmetric encryption, but less secure because the distribution of symmetric keys is not an easy operation.

Finally, we have the hybrid approach, where systems use asymmetric encryption to securely exchange a symmetric key (known as a session key). Once both parties have the symmetric key, they switch to the faster symmetric encryption for the rest of their communication. This method leverages the security of asymmetric encryption for key distribution and the efficiency of symmetric encryption for data transfer. The most famous example protocol of hybrid key exchange is TLS. This protocol uses hybrid schemes to establish secure connections over the Internet. The initial handshake might involve asymmetric methods to set up a secure channel, after which data is encrypted using symmetric encryption.

The benefits of (classical) key hybridization include, on one side, an enhancement in security, as com-

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

bining multiple encryption methods reduces the risk of the entire communication being compromised if one method is broken, and, on the other side, an increase in the efficiency and the scalability, allowing for the encryption of large volumes of data efficiently and making it practical for everyday use in a variety of applications, ranging from secure web browsing to mobile phone traffic encryption, secure file transfer protocols, confidential communications or any instance where secure data transmission is necessary.

### Hybridization of QKD and PQC

Following the same philosophy as in classical key hybridization, it is possible to hybridize key material for native quantum safe technology. The hybridization of key material between Quantum Key Distribution (QKD) and PQC represents a pioneering approach in the evolution of cryptographic security, combining the robustness of quantum-resistant algorithms with the inherent security features of quantum mechanics.

This hybridization sometimes just refers to the usage of PQC key encapsulation mechanisms and signatures to authenticate and secure the classical channel that is needed to perform QKD [124, 125, 126], but further ideas of combining the QKD key with some other PQC ones into a single key using a Key Derivation Function (KDF) — which can be for instance the combination of a hash function and a XOR mask [127, 128] — have been proposed. The main aspects of quantum-secure Hybridization Key Material are:

- *Different Cryptographic Systems*: hybridization often involves keys from both classical and quantum cryptographic systems. For example, combining QKD, which provides theoretically secure key exchange based on the principles of quantum mechanics, with PQC algorithms, which are designed to be secure against attacks from CRQCs.
- *Multi-Layer Security*: by using keys from different systems, hybridization aims to create a multi-layered security approach where the breach of one layer does not compromise the overall security. This is analogous to having multiple lines of defense in a security system.
- *Enhanced Robustness*: hybridization increases the robustness of cryptographic systems against future technological advances that might break current encryption methods. For example, if future advancements in quantum computing render certain PQC algorithms vulnerable, the quantum-secure keys distributed via QKD might still safeguard the system.
- *Flexibility and Scalability*: hybridization allows for greater flexibility in deployment and scalability. Organizations can implement robust security measures that can be adapted to different operational needs and threat levels, depending on the sensitivity and value of the information being protected.

Hybridizing QKD and PQC is an emerging approach in cybersecurity aimed at blending the strengths of both technologies to create a robust defense against various types of cyber threats, including those from CRQCs. However, while QKD provides theoretically secure key exchange, it requires a direct or authenticated channel and is limited by infrastructure and distance constraints. PQC, on the other hand, offers more flexibility and is easier to integrate into existing digital infrastructures, but the existence of a quantum algorithm that is able to break its security is still an open question (although there is strong evidence that support that there is no such quantum algorithm).

Hybrid systems aim to leverage the unconditional security of QKD for key distribution and the practicality and scalability of PQC algorithms, ensuring long-term security against both classical and quantum threats. By combining QKD and PQC, the hybrid system can protect against both potential vulnerabilities in PQC algorithms (should future breakthroughs make them vulnerable to quantum attacks) and the technological and implementation limitations of QKD. Furthermore, the dual-layer security model provides redundancy, significantly enhancing security, which means that, even if one system is compromised, the other layer can still secure the communications.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

#### Challenges in Hybridization

There are several challenges worth to be considered in the hybridization of QKD and PQC.

- *Implementation Complexity*: integrating two fundamentally different technologies involves significant challenges, including compatibility issues, increased system complexity, and higher operational costs.
- *Performance Overheads*: the hybrid system might introduce latency or overhead due to the need to manage and operate two distinct cryptographic systems.
- *Infrastructure Requirements*: deploying QKD requires a physical infrastructure for quantum channels, which can be costly and logistically challenging.

Hybrid QKD/PQC systems represent a forward-thinking approach to cybersecurity, aiming to combine the best of quantum-safe and quantum-proof technologies to secure communications against both current and future threats. This approach is particularly pertinent, as we are moving closer to the quantum era.

### 9.1.3. Quantum Key Distribution

QKD is a cryptographic method that uses quantum mechanics to generate and share an encryption key between two parties in a way that it is intrinsically secure from eavesdropping. QKD exploits the fundamental properties of quantum mechanics, such as the uncertainty principle and the no-cloning theorem [129], to ensure that any attempt at eavesdropping can be detected. The most common form of QKD, the BB84 protocol [130] introduced by Bennett and Brassard in 1984, involves encoding of bits on the polarization states of photons and securely distributing of qubits over a quantum channel. Any interception attempt disturbs these quantum states, revealing the presence of an eavesdropper.

QKD has advanced significantly from a theoretical concept to practical implementations in recent years, including deep integration with existing telecommunications infrastructure. Despite the challenge of maintaining the integrity of quantum states over long distances, recent technologies have enabled successful QKD over hundreds of kilometers using both fiber optic networks and free-space channels [131, 132, 133]. This makes QKD a robust solution for achieving long-term security, particularly valuable in an era where conventional encryption could potentially be compromised by quantum computing. As such, QKD is being increasingly tested in various applications from secure governmental communications to banking transactions, being potentially the way for a new standard in secure communications.

QKD will establish a secure link using free-space channels to generate secret keys for the IPsec protocol. This will be achieved through the implementation of the well-established BB84 protocol, which leverages the inherent security principles of quantum mechanics.

The BB84 protocol relies on the transmission of single photons encoded with specific polarization states. By comparing the transmitted and received states after successful detection, any eavesdropping attempt will introduce detectable errors due to the quantum uncertainty principle.

To ensure reliable key generation over the free-space channel, error correction algorithms (belief propagation decoding of LDPC codes) will be employed. This algorithm will identify and rectify errors introduced by noise and imperfections in the channel and detectors, and by the quantum nature of light itself, guaranteeing the integrity of the generated keys.

Furthermore, post-processing techniques will be implemented to estimate the security of the established keys and perform privacy amplification. This crucial step minimizes the information and eavesdropper could potentially gain from any residual errors, further enhancing the overall security of the QKD protocol.

Finally, the generated keys will be stored within the QKD devices themselves. Both the sender and receiver devices will possess identical keys, which will then be delivered to the key management component within the CCIPS agent via the ETSI QKD 004 standard API [134]. This ensures seamless integration with

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

the overall IPsec infrastructure and facilitates the secure distribution of quantum-safe keys for the IPsec encryption.

### 9.1.4. Remote Attestation

Integrity Verification will be included in the IPsec IKE-less architecture through Remote Attestation. The latter will be implemented by means of a Trust Manager (acting as a Verifier) in order to verify trustworthiness of the platform where CCIPS Agents are running. This process would occur before the establishment of Security Association (SA) and the negotiation of cryptographic parameters, and periodically upon request from the Trust Manager. The Attestation Agent is in charge of generating and signing the quote, which is sent to the Trust Manager for verification. The Trust Manager and the Attestation Agent will be included in the overall architecture depicted in Figure 9.3.

A remote attestation protocol would need to accommodate PQ digital signature algorithms for signing the attestation measurement.

Assuming successful remote attestation, the CCIPS Agents can communicate through a secure communication channel protected by IPsec.

## 9.2. Requirements

This section depicts those requirements associated with the transition to quantum-secure algorithms and key management for the IPsec building block.

**Table 9.1:** IPsec IKE-less requirements

| Req. | Description |
|---|---|
| IPsec-01 | Quantum secure authentication SHOULD be provided between IPsec endpoints, without dependency on IKEv2 |
| IPsec-02 | The IPsec must support adoption of RFC-9061 in IKE-less mode as a replacement of IKEv2 for the potential attacks to classical algorithms by quantum computers |
| IPsec-03 | Keys must not be shared over non-quantum secure channels outside a security perimeter |
| IPsec-04 | The IPSec may use pre-shared keys to establish connections if the RFC-9061 interface is not available as a quantum secure alternative |
| IPsec-05 | The hybridization solution should combine keys generated by QKD systems, PQC KEM algorithms and classical algorithms to produce robust derived keys, that offer higher resistance than only one of the previous methods |
| IPsec-06 | The hybridization solution should be able to generate non-hybrid quantum secure keys in case of lack of one of the key sources (QKD or PQC) |
| IPsec-07 | The hybridization solution should support standardized interfaces to retrieve keys from QKD system (ETSI QKD 004), allowing different QKD manufactures, Quality of Service (QoS) and size for the keys |
| IPsec-08 | The hybridization solution must support a modular approach that will allow alternate between different PQC algorithms, without impact on the functionality and external interfaces of the solution (crypto agility) |

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

| | |
|---|---|
| IPsec-09 | The attestation solution should provide additional protection based on quantum secure mechanism in case of applying classical algorithms in hardware for attestation |
| IPsec-10 | The QKD solution should provide quantum secure keys based on ETSI QKD 004 interface to be used by IPsec as part of the IKEv2 replacement |

The requirements for remote attestation in the IPsec IKE-less architecture are explained in Table 9.2.

**Table 9.2:** Remote Attestation PQ requirements for IPsec IKE-less.

| Req. | Description |
|---|---|
| IPsecRA-01 | The Attestation Agent MUST support PQ digital signature algorithms |
| IPsecRA-02 | Key generation, signature generation and signature verification speed MUST be considered to choose PQ algorithms |
| IPsecRA-03 | The Trust Manager MUST support PQ digital signature algorithms for verification |

## 9.3. Design

The design of the solution for IPsec transition is fundamentally based on the concept established by RFC-9061 [1] (see Section 9.1.1), combined with a hybridization solution and seeks to avoid the use of legacy IKEv2.

The solution is depicted in Figure 9.3 where the main components are represented, and here detailed:

- **CCIPS**: including the controller and multiple agents (attached to IPsec engines) to manage and establish dynamically on-demand different IPsec tunnels on the network. It distributes the IPsec SA configuration as an alternative to IKEv2.
- **PQC/QKD Hybridization**: takes care of using one or more sources of raw key material to create the symmetric keys needed by the IPsec. The role of this enabler is similar to a key management system to collect, process and store keys from different sources, and provide them on demand to the agents.
- **QKD modules**: provide a source of quantum-secure keys to the Hybridization module, based on an alternative approach to PQC, using quantum mechanics principles.
- **Trust Manager**: coordinates the authentication and integrity verification of the software components included in the agents.
- **Attestation Agent**: is in charge of generating the quote containing the software measurement needed for remote attestation. The quote is digitally signed with a PQ algorithm.

The proposed solution works as it follows. First, when one or more IPsec connections are demanded for a communication, the CCIPS Controller contacts the Trust Manager to identify and authenticate the CCIPS Agents, before triggering the IPsec setup process. This process includes the remote attestation of each agent involved. Once validated, the CCIPS agents receive the request with a common keyID and contact their local hybridization modules to request the keys. The received keys are created on demand by leveraging and mixing one or more of the possible methods to generate and agree upon keys: the QKD module with the ETSI QKD 004 application interface, and/or the NIST KEM method. Man-in-The-Middle (MiTM) attacks in the KEM key agreement method can be avoided by sharing public keys generated in the previous attestation process by each agent. Once the Agent receives the keys, it adds them to the final configurations, populate the local IPsec databases, and activates the IPsec tunnel.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

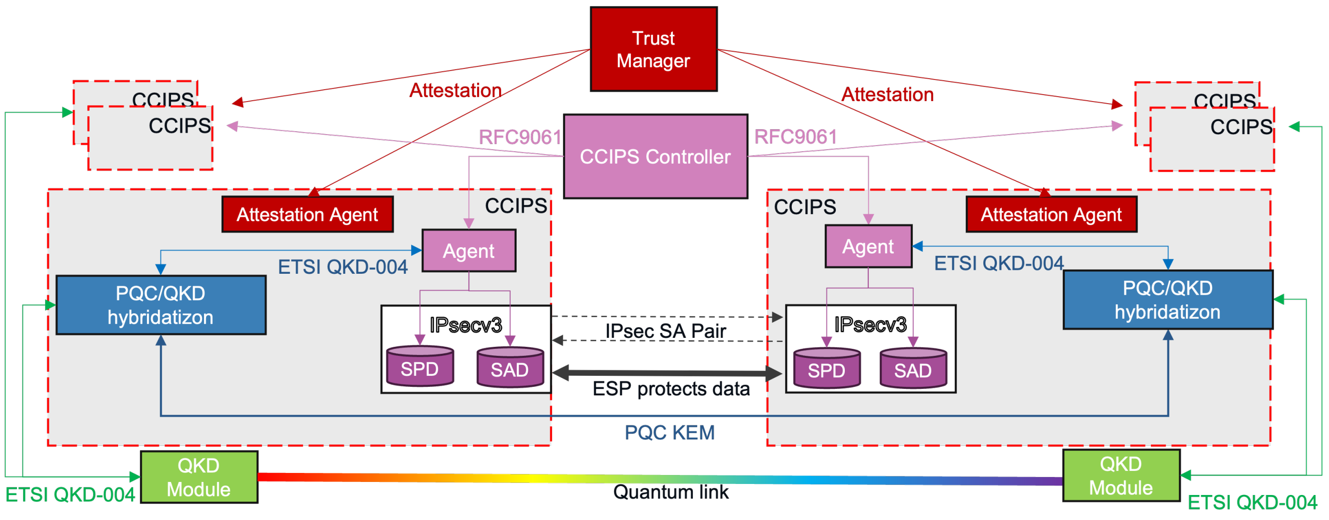


**Figure 9.3:** Overall design of the QKD/PQC hybrid IKE-less IPSec building block.

This initial design leverages as many standardized interfaces as possible to facilitate the transition to PQC: ETSI QKD 004 for key request and delivery, IETF RFC-9061 [1] for policies and control, and NIST selected KEM for PQC key agreements.

Our hybridization design follows the recommendations from NIST for cryptographic key generation [128]. There are essentially three different methods to derive one single key (the combined key) from multiple independent keys (the component keys): *concatenation*, *exclusive-oring* and *key-extraction*. The difference among those three methods is essentially the complexity of the key combination process and the relation between the min-entropies[1] of each component key and the combined key.

- For the *concatenation* method, the combined key $K$ can be derived from the component keys $K_1$ and $K_2$ as

  $$K = K_1 || K_2$$

  Here the sum of the min-entropies of the component keys $K_1$ and $K_2$ shall be equal or greater than the required min-entropy of the combined key $K$. Also, the size of the combined key will be the sum of the sizes of the component keys.
- For the *exclusive-oring* method, the combined key $K$ can be derived from the component keys $K_1$ and $K_2$ as

  $$K = K_1 \oplus K_2 \oplus \bar{0}$$

  where $\bar{0}$ refers to a string of zeros of the same length as the component keys. This method requires that at least one of the component keys has equal or greater min-entropy than the min-entropy required for the combined key. Also, the length of each component key shall be equal the length of the combined key.
- The *key extraction* process can be written as

  $$K = T(\text{HMAC-}hash(salt, K_1 || K_2), kLen)$$

  where $hash$ is a hash function, $salt$ is a secret or non-secret value with length $\geq 0$ (and must be known by all entities using this method), $kLen$ is the desired length of the combined key, and $T$ is the truncation function. The length of the output block of the hash function used with HMAC shall be at least $kLen$ bits, the required bit length for the combined key. The sum of the min-entropies of the

[1]When we refer to min-entropy of a key, we are referring to the min-entropy of the method used to generate or establish the key.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

component keys shall be equal to or greater than the min-entropy of the combined key and should be at least twice that amount of min-entropy.

These three different methods will be implemented in the hybridization module, giving the possibility to choose the one that fits better with the characteristics or conditions of the key material given by PQC and QKD. The hybridization module will use, whenever possible, the component keys coming from QKD and PQC to create a combined key through one of the three hybridization methods enumerated above. However, if the component key given from one source is not available, the combined key will be the component key obtained from the available source. This could be the case, for example, if the noise of the QKD system is above the security threshold. It is worth to mention that this hybridization design is not limited to the combination of one quantum key and one post-quantum key. The component keys $K_1$ and $K_2$ can be obtained through classical, quantum or post-quantum methods, and the combined key $K$ will remain secure as long as at least one component key remains secure.

This transition exercise addresses the security of network transmission based on the *store-now-decrypt-later* threat model. Regarding the remote attestation, the Attestation agents will adopt PQ algorithms to sign the attestation evidence generated by the physical (i.e., classical) TPM. Currently, a pure PQ physical TPM has not been developed yet. Therefore, PQ integrity verification through remote attestation in the IKE-less IPsec architecture will be integrated by means of a driver extension at kernel level to wrap the quote (already signed with classical algorithms) with an additional PQ digital signature.

The TPM will also be in charge of securely storing the PQ keys used by the Attestation agent to sign the quote. In this way, an initial experimental step will be made towards the development of a pure PQ TPM, which is out of the scope of the QUBIP project. SLH-DSA has been chosen as PQ digital signature algorithm during the remote attestation protocol.

The PQ keys will be generated by using the kernel driver extension, following secure practices to maintain the integrity and confidentiality of the keys. In fact, such keys will be imported and stored in the TPM. During this process, the private key must be encrypted with a key that the TPM can understand, using an algorithm supported by the physical TPM, namely either RSA or ECC.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

# 10. Conclusions

The QUBIP partners have described in this deliverable their analysis and design for PQ transition of the building blocks needed in the use cases of the project. However, it should be noted that the PQ scenario is rapidly changing, so the content of this deliverable reflects the current SotA and the QUBIP partners will do their best to follow its evolution during the project implementation. Nonetheless, we think that this document provides useful considerations and hints for other people looking into transition to PQC.

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

# Bibliography

[1] R. Marin-Lopez, G. Lopez-Millan, and F. Pereniguez-Garcia, “A YANG Data Model for IPsec Flow Protection Based on Software-Defined Networking (SDN)”, RFC-9061, July 2021, DOI 10.17487/RFC9061

[2] P. Kampanakis, P. C. van Oorschot, S. A. Vanstone, and Q. Nguyen, “Post-Quantum LMS and SPHINCS+ Hash-Based Signatures for UEFI Secure Boot”, IACR Cryptology ePrint Archive, Report 2021/041, 2021, https://eprint.iacr.org/2021/041.pdf

[3] F. Campos, T. Kohlstadt, S. Reith, and M. Stöttinger, “LMS vs XMSS: Comparison of Stateful Hash-Based Signature Schemes on ARM Cortex-M4”, AFRICACRYPT 2020, Cairo (EG), July 2020, pp. 258–277, DOI 10.1007/978-3-030-51938-4_13

[4] CCN, “Recommendations for a safe post-quantum transition”, December 2022, https://www.ccn.cni.es/index.php/eu/docman/documentos-publicos/boletines-pytec/499-ccn-tec-009-recomendaciones-transicion-postcuantica-segura-english/file

[5] ANSSI, “ANSSI views on the Post-Quantum Cryptography transition”, January 2022, https://cyber.gouv.fr/en/publications/anssi-views-post-quantum-cryptography-transition

[6] ANSSI, “Follow up position paper on post-quantum cryptography”, December 2023, https://cyber.gouv.fr/en/publications/follow-position-paper-post-quantum-cryptography

[7] BSI, “Cryptographic Mechanisms: Recommendations and Key Lengths. Technical Guideline TR-02102”, February 2024, https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.html

[8] NIST Computer Security Resource Center, “Post-Quantum Cryptography”, January 2017, https://csrc.nist.gov/projects/post-quantum-cryptography

[9] NSA, “Announcing the Commercial National Security Algorithm Suite 2.0”, 2022, https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF

[10] E. Alkim, J. W. Bos, L. Ducas, K. Easterbrook, B. LaMacchia, P. Longa, I. Mironov, M. Naehrig, V. Nikolaenko, C. Peikert, A. Raghunathan, and D. Stebila, “FrodoKEM: Learning With Errors Key Encapsulation. Preliminary Draft Standards”, March 2023, https://frodokem.org/files/FrodoKEM-ISO-20230314.pdf

[11] M. R. Albrecht, D. J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Misoczki, R. Niederhagen, K. G. Paterson, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, C. J. Tjhai, M. Tomlinson, and W. Wang, “Classic McEliece: Conservative code-based cryptography: Cryptosystem specification”, October 2022, https://classic.mceliece.org/mceliece-spec-20221023.pdf

[12] S. O. Bradner, “Key words for use in RFCs to Indicate Requirement Levels”, RFC-2119, March 1997, DOI 10.17487/RFC2119

[13] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, “Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU v1.2”, October 2020, https://falcon-sign.info

[14] T. Reddy, “Post-Quantum Cryptography Recommendations for Internet Applications”, Internet Draft RFC, March 2024, https://datatracker.ietf.org/doc/draft-reddy-uta-pqc-app/02/

[15] CA/Browser Forum, “Ballot SC22 – Reduce Certificate Lifetimes (v2)”, https://cabforum.org/2019/09/10/ballot-sc22-reduce-certificate-lifetimes-v2/

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

[16] BSI, "Quantum-safe cryptography – fundamentals, current developments and recommendations", 2022, https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/Brochure/quantum-safe-cryptography.html

[17] D. McGrew, M. Curcio, and S. Fluhrer, "Leighton-Micali Hash-Based Signatures", RFC-8554, April 2019, DOI 10.17487/RFC8554

[18] A. Huelsing, D. Butin, S.-L. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC 8391, May 2018, DOI 10.17487/RFC8391

[19] K. Bashiri, S. Fluhrer, S.-L. Gazdag, D. V. Geest, and S. Kousidis, "Internet X.509 Public Key Infrastructure: Algorithm Identifiers for HSS and XMSS", Internet Draft RFC, May 2024, https://datatracker.ietf.org/doc/draft-ietf-lamps-x509-shbs/00/

[20] NSA, "The commercial national security algorithm suite 2.0 and quantum computing FAQ", September 2022, https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/0/CSI_CNSA_2.0_FAQ_.PDF

[21] A. Huelsing, D. Butin, S.-L. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: eXtended Merkle Signature Scheme", RFC-8391, May 2018, DOI 10.17487/RFC8391

[22] GSMA Association, "Post Quantum Cryptography – Guidelines for Telecom Use Cases", 2024, https://www.gsma.com/newsroom/wp-content/uploads//PQ.03-Post-Quantum-Cryptography-Guidelines-for-Telecom-Use-v1.0.pdf

[23] GlobalSign, "Examples of Quantum-Safe X.509 Certificates", https://github.com/globalsign/example-pq-safe-x509

[24] Open Quantum Safe project, "Open Quantum Safe: software for the transition to quantum-resistant cryptography", https://openquantumsafe.org/

[25] IETF Hackathon, "IETF Hackathon – PQC Certificates", https://github.com/IETF-Hackathon/pqc-certificates

[26] Chromium Blog, "Protecting Chrome Traffic with Hybrid Kyber KEM", https://blog.chromium.org/2023/08/protecting-chrome-traffic-with-hybrid.html

[27] The Cloudflare Blog, "The state of the post-quantum Internet", https://blog.cloudflare.com/pq-2024

[28] F. Bene and A. Kiss, "Public Key Infrastructure in the Post-Quantum Era", IEEE 17th Int. Symposium on Applied Computational Intelligence and Informatics (SACI), 2023, pp. 77–82, DOI 10.1109/SACI58269.2023.10158562

[29] C. Wang, W. Xue, and J. Wang, "Integration of Quantum-Safe Algorithms into X.509v3 Certificates", IEEE 3rd Int. Conf. on Electronic Technology, Communication and Information (ICETCI), Changchun (China), 2023, DOI 10.1109/ICETCI57876.2023.10176713

[30] N. Bindel, B. Hale, D. Connolly, and F. Driscoll, "Hybrid signature spectrums", Internet Draft RFC, May 2024, https://datatracker.ietf.org/doc/draft-ietf-pquip-hybrid-signature-spectrums/00/

[31] C. Bonnell, J. Gray, D. Hook, T. Okubo, and M. Ounsworth, "A Mechanism for Encoding Differences in Paired Certificates", Internet Draft RFC, January 2024, https://datatracker.ietf.org/doc/draft-bonnell-lamps-chameleon-certs/03/

[32] M. Ounsworth, J. Gray, M. Pala, and J. Klaußner, "Composite ML-DSA for use in Internet PKI", Internet Draft RFC, March 2024, https://datatracker.ietf.org/doc/draft-ounsworth-pq-composite-sigs/13/

[33] J. Massimo, P. Kampanakis, S. Turner, and B. Westerbaan, "Internet X.509 Public Key Infrastructure: Algorithm Identifiers for ML-DSA", Internet Draft RFC, February 2024, https://datatracker.ietf.org/doc/draft-ietf-lamps-dilithium-certificates/03/

[34] NIST Computer Security Resource Center, "Post-quantum cryptography: Security (evaluation criteria)", 2017, https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria)

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

[35] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: Practical stateless hash-based signatures", EUROCRYPT 2015, Sofia (BG), April 2015, pp. 368–397, DOI 10.1007/978-3-662-46800-5_15

[36] D. Cooper, D. Apon, Q. Dang, M. Davidson, M. Dworkin, and C. Miller, "Recommendation for Stateful Hash-Based Signature Schemes", NIST SP800-208, October 2020, DOI 10.6028/NIST.SP.800-208

[37] A. Hülsing, C. Busold, and J. Buchmann, "Forward secure signatures on smart cards", 19th Int. Conf. on Selected Areas in Crytography, Windsor (CA), August 2012, pp. 66–80, DOI 10.1007/978-3-642-35999-6_5

[38] S. Ghosh, R. Misoczki, and M. R. Sastry, "Lightweight Post-Quantum-Secure Digital Signature Approach for IoT Motes", IACR Cryptology ePrint Archive, Paper 2019/122, 2019, https://eprint.iacr.org/2019/122

[39] V. B. Y. Kumar, N. Gupta, A. Chattopadhyay, M. Kaspert, C. Krauß, and R. Niederhagen, "Post-quantum secure boot", 23rd Conf. on Design, Automation and Test in Europe, Grenoble (FR), 2020, pp. 1582—-1585

[40] S. Fluhrer and P. Kampanakis, "LMS vs XMSS: A Comparison of the Stateful Hash-Based Signature Proposed Standards", IACR Cryptology ePrint Archive, Paper 2017/349, 2017. http://eprint.iacr.org/2017/349

[41] L. K. Grover, "A fast quantum mechanical algorithm for database search", 28th Annual ACM Symposium on Theory of Computing, Philadelphia (PA, USA), 1996, pp. 212—219, DOI 10.1145/237814.237866

[42] M.-Z. Mina and E. Simion, "Threats to modern cryptography: Grover's algorithm", Preprints, September 2020, https://www.preprints.org/manuscript/202009.0677/v1

[43] D. J. Bernstein and T. Lange, "Post-quantum cryptography", Nature, vol. 549, September 2017, pp. 188–194, DOI 10.1038/nature23461

[44] NIST, "FIPS 204, Module-Lattice-Based Digital Signature Standard – Initial Public Draft", 2023, https://doi.org/10.6028/NIST.FIPS.204.ipd

[45] P. Urien, "Towards internet of secure elements", IEEE 19th Annual Consumer Communications & Networking Conf. (CCNC), Las Vegas (NV, USA), January 2022, pp. 949–950, DOI 10.1109/CCNC49033.2022.9700663

[46] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", RFC-8446, August 2018, DOI 10.17487/RFC8446

[47] M. Vauclair, "Secure element", Encyclopedia of Cryptography and Security (H. C. A. van Tilborg and S. Jajodia, eds.), pp. 1115–1116, Springer, 2011, DOI 10.1007/978-1-4419-5906-5_303

[48] P. Urien, "Internet of secure elements concepts and applications", 7th Int. Conf. on Mobile And Secure Services (MobiSecServ), Gainesville (FL, USA), March 2022, pp. 1–6, DOI 10.1109/MobiSecServ50855.2022.9727207

[49] NIST, "FIPS 140-3, Security Requirements for Cryptographic Modules", 2019, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-3.pdf

[50] NIST, "FIPS 140-2, Security Requirements for Cryptographic Modules", 2001, https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf

[51] GlobalPlatform Security Task Force, "Secure Channel Protocol '03' – Amendment D v1.2", 2020, https://globalplatform.org/specs-library/secure-channel-protocol-03-amendment-d-v1-2/

[52] ST Microelectronics, "STM32F4 Series", https://www.st.com/en/microcontrollers-microprocessors/stm32f4-series.html

[53] Digilent, "Genesys 2", https://digilent.com/reference/programmable-logic/genesys-2/start

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

[54] ARM, “ARM IHI 0022 – AMBA AXI Protocol”, 2023, https://documentation-service.arm.com/static/651c285c15583d1bff972f94

[55] Xilinx, “Zynq UltraScale+ MPSoC ZCU104 Evaluation Kit”, 2024, https://www.xilinx.com/products/boards-and-kits/zcu104.html

[56] Trusted Firmware project, “Trusted Firmware-M Documentation”, https://tf-m-user-guide.trustedfirmware.org/

[57] Trusted Firmware project, “OP-TEE Documentation”, https://optee.readthedocs.io/en/latest/

[58] The OpenSSL project, “Cryptography and SSL/TLS Toolkit”, https://www.openssl.org

[59] The OpenSSL project, “provider OpenSSL 3.2 Manpage”, January 2024, https://www.openssl.org/docs/man3.2/man7/provider.html

[60] F. Driscoll and M. Parsons, “Terminology for Post-Quantum Traditional Hybrid Schemes”, Internet Draft RFC, May 2024, https://datatracker.ietf.org/doc/draft-ietf-pquip-pqt-hybrid-terminology/03/

[61] Mozilla, “Network Security Services (NSS)”, https://firefox-source-docs.mozilla.org/security/nss/index.html

[62] Mozilla, “Netscape Portable Runtime (NSPR)”, https://firefox-source-docs.mozilla.org/nspr/index.html

[63] E. Rescorla and T. Dierks, “The Transport Layer Security (TLS) Protocol Version 1.2”, RFC-5246, August 2008, DOI 10.17487/RFC5246

[64] J. Schaad, B. C. Ramsdell, and S. Turner, “Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification”, RFC-8551, April 2019, DOI 10.17487/RFC8551

[65] S. Turner, “Asymmetric Key Packages”, RFC-5958, August 2010, DOI 10.17487/RFC5958

[66] K. Moriarty, M. Nyström, S. Parkinson, A. Rusch, and M. Scott, “PKCS#12: Personal Information Exchange Syntax v1.1”, RFC-7292, July 2014, DOI 10.17487/RFC7292

[67] OASIS PKCS#11 TC, “PKCS#11 specification version 3.1”, July 2023, https://docs.oasis-open.org/pkcs11/pkcs11-spec/v3.1/os/pkcs11-spec-v3.1-os.html

[68] Trusted Firmware project, “MbedTLS library”, https://github.com/Mbed-TLS/mbedtls

[69] Trusted Firmware project, “Trusted firmware, Open Source Secure Software”, https://www.trustedfirmware.org/

[70] Trusted Firmware project, “Trusted Firmware-A Documentation”, https://trustedfirmware-a.readthedocs.io/en/latest/

[71] The OpenSSL project, “Capabilities at provider-base OpenSSL 3.2 Manpage”, January 2024, https://www.openssl.org/docs/man3.2/man7/provider-base.html#CAPABILITIES

[72] NIST CSRC, “PQC – API Notes”, March 2017, https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/archive/api-march2017.pdf

[73] The OpenSSL project, “Apache License Version 2.0”, https://www.openssl.org/source/apache-license-2.0.txt

[74] S. Klabnik and C. Nichols, “The Rust Programming Language, 2nd Edition”, No Starch Press, 2023, ISBN: 9781718503106. https://doc.rust-lang.org/book/ch19-06-macros.html

[75] Mozilla, “Mozilla Public License Version 2.0”, https://hg.mozilla.org/projects/nss/file/tip/COPYING

[76] Mozilla, “Oxidation: Integrating Rust into Mozilla’s Codebase”, November 2020, https://wiki.mozilla.org/Oxidation

[77] Fedora, “Fedora Project”, https://fedoraproject.org

[78] Flatpak, “The future of apps on Linux”, https://flatpak.org/

[79] Open Quantum Safe project, “NIST algorithms in liboqs”, https://openquantumsafe.org/liboqs/algorithms/

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

[80] Fedora, “Rawhide”, https://docs.fedoraproject.org/en-US/releases/rawhide/

[81] Mozilla, “Firefox browsers”, https://www.mozilla.org/en-GB/firefox/browsers/

[82] A. Preukschat and D. Reed, “Self-Sovereign Identity – Decentralized digital identity and verifiable credentials”, Manning, 2021, ISBN: 9781617296598. https://www.manning.com/books/self-sovereign-identity

[83] W3C, “Decentralized Identifiers (DIDs) v1.0. Core architecture, data model, and representations. W3C Recommendation”, 2022, https://www.w3.org/TR/did-core/

[84] W3C, “DID Specification Registries. The interoperability registry for Decentralized Identifiers. W3C Group Note”, 2024, https://www.w3.org/TR/did-spec-registries/

[85] IOTA Foundation, “Digital Identity”, 2024, https://www.iota.org/solutions/digital-identity

[86] Privado ID, “Digital Identity”, 2024, https://www.privado.id/

[87] W3C, “did:web Method Specification”, 2023, https://w3c-ccg.github.io/did-method-web/

[88] W3C, “The did:key Method Specification. A DID Method for Static Cryptographic Keys”, 2022, https://w3c-ccg.github.io/did-method-key/

[89] W3C, “Verifiable Credentials Data Model v2.0. W3C Candidate Recommendation Draft”, 2024, https://www.w3.org/TR/vc-data-model-2.0/

[90] W3C, “Bitstring Status List v1.0: Privacy-preserving status information for Verifiable Credentials. W3C Working Draft”, 2024, https://www.w3.org/TR/vc-bitstring-status-list/

[91] J. Camenisch and A. Lysyanskaya, “A signature scheme with efficient protocols”, Security in Communication Networks, Amalfi (IT), September 2002, pp. 268–289, DOI 10.1007/3-540-36413-7_20

[92] IOTA Foundation, “IOTA Identity library”, 2024, https://github.com/iotaledger/identity.rs

[93] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”, SIAM Journal on Computing, vol. 26, October 1997, pp. 1484–1509, DOI 10.1137/S0097539795293172

[94] J. Camenisch, M. Drijvers, and A. Lehmann, “Anonymous attestation using the strong Diffie-Hellman assumption revisited”, TRUST-2016 – 9th Int. Conf. on Trust and Trustworthy Computing, Vienna (AT), August 2016, pp. 1–20, DOI 10.1007/978-3-319-45572-3_1

[95] C. Jeudy, A. Roux-Langlois, and O. Sanders, “Lattice signature with efficient protocols, application to anonymous credentials”, IACR Cryptology ePrint Archive, Paper 2022/509, 2022, https://eprint.iacr.org/2022/509

[96] Q. Lai, C. Chen, F.-H. Liu, A. Lysyanskaya, and Z. Wang, “Lattice-based commit-transferrable signatures and applications to anonymous credentials”, IACR Cryptology ePrint Archive, Paper 2023/766, 2023, https://eprint.iacr.org/2023/766

[97] J. Bootle, V. Lyubashevsky, N. K. Nguyen, and A. Sorniotti, “A framework for practical anonymous credentials from lattices”, IACR Cryptology ePrint Archive, Paper 2023/560, 2023, https://eprint.iacr.org/2023/560

[98] J. Bootle, V. Lyubashevsky, N. K. Nguyen, and A. Sorniotti, “A framework for practical anonymous credentials from lattices”, CRYPTO 2023, Santa Barbara (CA, USA), August 2023, pp. 384–417, DOI 10.1007/978-3-031-38545-2_13

[99] B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang, “Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions”, ASIACRYPT 2016, Hanoi (VN), December 2016, pp. 373–403, DOI 10.1007/978-3-662-53890-6_13

[100] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, “CRYSTALS–Kyber: A CCA-Secure Module-Lattice-Based KEM”, IEEE European Symposium on Security and Privacy (EuroS&P), London (UK), 2018, pp. 353–367, DOI 10.1109/EuroSP.2018.00032

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

[101] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS–Dilithium: A lattice-based digital signature scheme", IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2018, February 2018, pp. 238—268, DOI 10.13154/tches.v2018.i1.238-268

[102] V. Lyubashevsky, N. K. Nguyen, and M. Plançon, "Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general", CRYPTO 2022, Santa Barbara (CA, USA), August 2022, pp. 71–101, DOI 10.1007/978-3-031-15979-4_3

[103] M. Ajtai, "Generating hard instances of lattice problems", Electronic Colloquium on Computational Complexity, vol. TR96, January 1996. https://eccc.weizmann.ac.il/report/1996/007/download

[104] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions", IACR Cryptology ePrint Archive, Paper 2007/432, 2007, https://eprint.iacr.org/2007/432

[105] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller", IACR Cryptology ePrint Archive, Paper 2011/501, 2011, https://eprint.iacr.org/2011/501

[106] M. R. Albrecht, V. Cini, R. W. F. Lai, G. Malavolta, and S. A. Thyagarajan, "Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable", IACR Cryptology ePrint Archive, Paper 2022/941, 2022, https://eprint.iacr.org/2022/941

[107] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials", CRYPTO 2002, Santa Barbara (CA, USA), August 2002, pp. 61–76, DOI 10.1007/3-540-45708-9_5

[108] J. Camenisch, M. Kohlweiss, and C. Soriente, "An accumulator based on bilinear maps and efficient revocation for anonymous credentials", PKC 2009, Irvine (CA, USA), March 2009, pp. 481–500, DOI 10.1007/978-3-642-00468-1_27

[109] J. Camenisch, M. Kohlweiss, and C. Soriente, "Solving revocation with efficient update of anonymous credentials", SCN 2010, Amalfi (IT), September 2010, pp. 454–471, DOI 10.1007/978-3-642-15317-4_28

[110] Open Quantum Safe project, "liboqs", 2024, https://github.com/open-quantum-safe/liboqs

[111] A. Pino, D. Margaria, and A. Vesco, "On PQ/T hybrid verifiable credentials and presentations to build trust in IoT systems", 9th Int. Conf. on Smart and Sustainable Technologies (SpliTech 2024), Split (HR), June 2024. *to be published*

[112] N. Bindel, U. Herath, M. McKague, and D. Stebila, "Transitioning to a quantum-resistant public key infrastructure", Cryptology ePrint Archive, Paper 2017/460, 2017, https://eprint.iacr.org/2017/460

[113] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: a new high speed public key cryptosystem", presented at the rump session of Crypto'96, 1996, https://web.securityinnovation.com/hubfs/files/ntru-orig.pdf

[114] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem", ANTS-III – 3rd Int. Symp. on Algorithmic Number Theory, Portland (OR, USA), June 1998, pp. 267–288, DOI 10.1007/BFb0054868

[115] G. Alagic, 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, D. Smith-Tone, and D. Apon, "Status report on the third round of the NIST post-quantum cryptography standardization process", July 2022, DOI 10.6028/NIST.IR.8413

[116] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices", Designs, Codes and Cryptography, vol. 75, June 2015, pp. 565–599, DOI 10.1007/s10623-014-9938-4

[117] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography", 37th An-

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

nual ACM Symposium on Theory of Computing, Baltimore (MD, USA), 2005, pp. 84—93, DOI 10.1145/1060590.1060603

[118] NIST, “FIPS 203, Module-Lattice-Based Key-Encapsulation Mechanism Standard – Initial Public Draft”, 2023, https://doi.org/10.6028/NIST.FIPS.203.ipd

[119] K. Seo and S. Kent, “Security Architecture for the Internet Protocol”, RFC-4301, December 2005, DOI 10.17487/RFC4301

[120] IANA, “Internet Key Exchange Version 2 (IKEv2) Parameters”, 2024, https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml

[121] NIST, “Post-Quantum Cryptography FAQs on Transition and Migration”, https://csrc.nist.gov/Projects/post-quantum-cryptography/faqs#xisl

[122] C. Kaufman, P. E. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, “Internet Key Exchange Protocol Version 2 (IKEv2)”, RFC-7296, October 2014, DOI 10.17487/RFC7296

[123] Y. Nir, T. Kivinen, P. Wouters, and D. Migault, “Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2)”, RFC-8247, September 2017, DOI 10.17487/RFC8247

[124] D. Marchsreiter and J. Sepúlveda, “A PQC and QKD Hybridization for Quantum-Secure Communications”, 26th Euromicro Conf. on Digital System Design (DSD), Durres (AL), September 2023, pp. 545–552, DOI 10.1109/DSD60849.2023.00081

[125] L.-J. Wang, K.-Y. Zhang, J.-Y. Wang, J. Cheng, Y.-H. Yang, S.-B. Tang, D. Yan, Y.-L. Tang, Z. Liu, Y. Yu, Q. Zhang, and J.-W. Pan, “Experimental Athentication of Quantum Key Distribution with Post-Quantum Cryptography”, npj Quantum Information, vol. 7, May 2021, p. 67, DOI 10.1038/s41534-021-00400-7

[126] A. Prakasan, K. Jain, and P. Krishnan, “Authenticated-encryption in the quantum key distribution classical channel using post-quantum cryptography”, 6th Int. Conf. on Intelligent Computing and Control Systems (ICICCS), 2022, pp. 804–811, DOI 10.1109/ICICCS53718.2022.9788239

[127] M. Geitz, R. Döring, and R.-P. Braun, “Hybrid QKD & PQC Protocols Implemented in the Berlin OpenQKD Testbed”, 8th Int. Conf. on Frontiers of Signal Processing (ICFSP), 2023, pp. 69–74, DOI 10.1109/ICFSP59764.2023.10372894

[128] E. Barker, E. Barker, A. Roginsky, and R. Davis, “Recommendation for cryptographic key generation (revision 2)”, NIST SP800-133r2, June 2020, DOI 10.6028/NIST.SP.800-133r2

[129] B. Zhao, B. Liu, C. Wu, W. Yu, and I. You, “A tutorial on quantum key distribution”, 10th Int. Conf. on Broadband and Wireless Computing, Communication and Applications (BWCCA), November 2015, DOI 10.1109/bwcca.2015.77

[130] C. H. Bennett and G. Brassard, “Quantum cryptography: Public key distribution and coin tossing”, Theoretical Computer Science, vol. 560, December 2014, pp. 7—11, DOI 10.1016/j.tcs.2014.05.025

[131] Y. Liu, W.-J. Zhang, C. Jiang, J.-P. Chen, C. Zhang, W.-X. Pan, D. Ma, H. Dong, J.-M. Xiong, C.-J. Zhang, H. Li, R.-C. Wang, J. Wu, T.-Y. Chen, L. You, X.-B. Wang, Q. Zhang, and J.-W. Pan, “Experimental Twin-Field Quantum Key Distribution over 1000 km Fiber Distance”, Physical Review Letters, vol. 130, May 2023, p. 210801, DOI 10.1103/PhysRevLett.130.210801

[132] R. Sax, A. Boaron, G. Boso, S. Atzeni, A. Crespi, F. Grünenfelder, D. Rusca, A. Al-Saadi, D. Bronzi, S. Kupijai, H. Rhee, R. Osellame, and H. Zbinden, “High-speed integrated QKD system”, Photonics Research, vol. 11, June 2023, pp. 1007–1014, DOI 10.1364/PRJ.481475

[133] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. Pereira, M. Razavi, J. S. Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden, “Advances in Quantum Cryptography”, Advances in Optics and Photonics, vol. 12, December 2020, p. 1012, DOI 10.1364/AOP.361502

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

[134] ETSI, “Quantum Key Distribution (QKD); Application Interface”, 2020, https://www.etsi.org/deliver/etsi_gs/QKD/001_099/004/02.01.01_60/gs_qkd004v020101p.pdf

[135] Cisco Systems, “A full-featured implementation of of the LMS and HSS Hash Based Signature Schemes from draft-mcgrew-hash-sigs-07”, https://github.com/cisco/hash-sigs

[136] SPHINCS+ Team, “SPHINCS+ Implementation”, https://github.com/sphincs/sphincsplus

[137] A. Huelsing, D. Butin, S.-L. Gazdag, J. Rijneveld, and A. Mohaisen, “XMSS Reference Code”, https://github.com/XMSS/xmss-reference

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

## A. DID Document, VC and VP Data Models

The following paragraphs present examples of a DID document, a VC, and a VP in JavaScript Object Notation (JSON) format compliant with the W3C specified data models [83, 89].

Here is an example of DID document, refer to [83] for detailed explanation of all fields.

```
"id": "did:method_name:method_specific_id",
"authentication": [{
    "id": "did:method_name:method_spec_id#keys-1",
    "type": "JsonWebKey2020",
    "controller": "did:method_name:method_spec_id",
    "publicKeyJwk": { .. encoded public key .. }
    },{
    "id": "did:method_name:method_spec_id#keys-2",
    "type": "JsonWebKey2020",
    "controller": "did:method_name:method_spec_id",
    "publicKeyJwk": { .. encoded public key .. }
}]
```

Here is an example of Verifiable Credential, refer to [89] for detailed explanation of all fields.

```
"@context": ["https://www.w3.org/2018/credentials/v1"],
"id": "https://address/credentials/1",
"type": ["VerifiableCredential"],
"issuer": "did:method_name:method_specific_id",
"issuanceDate": " .. date and time ..  ",
"expirationDate": " .. date and time ..  ",
"credentialSubject": {
    "id": "did:method_name:method_specific_id",
    .. properties to describe the identity ..
},
"credentialStatus": {
    "id": "https://address/credentials/status/#",
    "type": "BitstringStatusListEntry",
    "statusPurpose": "revocation",
    "statusListIndex": " .. index .. ",
}
"proof": {
    "type": " .. type of signature .. ",
    "created": " .. date and time .. ",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:method_name:method_specific_id#fragment",
    "proofValue": " .. the encoded signature .. "
}
```

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

Here is an example of Verifiable Presentation, refer to [89] for detailed explanation of all fields.

```
"@context": ["https://www.w3.org/2018/credentials/v1"],
"type": "VerifiablePresentation",
"verifiableCredential": [{ .. the VC .. }],
"proof": {
    "type": " .. type of signature .. ",
    "created": " .. date and time .. ",
    "proofPurpose": "authentication",
    "verificationMethod": "did:method_name:method_specific_id#fragment",
    "challenge": " .. challenge from the Verifier .. ",
    "proofValue": " .. the encoded signature .. "
}
```

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

## B. Experimental Evaluation of NIST Selected PQ Signature Algorithms

**Table B.1:** Performance of liboqs implementations of NIST selected PQ digital signature algorithms on a Intel® Core™ i7-1255U 1.70 GHz, Intel Turbo Boost disabled, RAM 24.0 GB 3200 MHz; keypair and signature generation and signature verification values are the average over 1000 runs.

| Algorithm | NIST Security Level | Public Key Size JWK encoded (Bytes) | Sign Size (Bytes) | Keypair Generation (ms) | Sign Generation (ms) | Sign Verification (ms) |
|---|---|---|---|---|---|---|
| Ed25519 | - | 145 | 64 | 0,042 | 0,042 | 0,070 |
| FALCON 512 | 1 | 1291 | 752 | 13,361 | 0,685 | 0,086 |
| SLH-DSA-SHA2-128f | 1 | 147 | 17088 | 2,745 | 61,541 | 3,686 |
| SLH-DSA-SHA2-128s | 1 | 147 | 7856 | 167,310 | 1263,700 | 1,331 |
| SLH-DSA-SHAKE-128f | 1 | 148 | 17088 | 3,802 | 86,079 | 5,023 |
| SLH-DSA-SHAKE-128s | 1 | 148 | 7856 | 238,820 | 1807,600 | 1,848 |
| ML-DSA-44 | 2 | 1845 | 2420 | 0,252 | 0,694 | 0,151 |
| ML-DSA-65 | 3 | 2698 | 3293 | 0,373 | 1,028 | 0,241 |
| SLH-DSA-SHA2-192f | 3 | 168 | 35664 | 3,892 | 100,110 | 5,375 |
| SLH-DSA-SHA2-192s | 3 | 168 | 16224 | 241,470 | 2250,100 | 1,925 |
| SLH-DSA-SHAKE-192f | 3 | 169 | 35664 | 5,575 | 141,390 | 7,815 |
| SLH-DSA-SHAKE-192s | 3 | 169 | 16224 | 348,670 | 3124,700 | 2,613 |
| FALCON 1024 | 5 | 2487 | 1462 | 40,792 | 1,140 | 0,168 |
| ML-DSA-87 | 5 | 3551 | 4595 | 0,503 | 1,263 | 0,405 |
| SLH-DSA-SHA2-256f | 5 | 190 | 49856 | 10,001 | 204,800 | 5,490 |
| SLH-DSA-SHA2-256s | 5 | 190 | 29792 | 159,610 | 1990,700 | 2,796 |
| SLH-DSA-SHAKE-256f | 5 | 191 | 49856 | 14,513 | 291,390 | 7,824 |
| SLH-DSA-SHAKE-256s | 5 | 191 | 29792 | 230,200 | 2766,500 | 3,815 |

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

# C. BLNS Framework

This appendix provides a high-level description of the BLNS framework [97, 98], that has been selected and adopted in QUBIP as a building block for the PQ Anonymous Verifiable Credentials system (see Chapter 8). In detail, after recalling the mathematical notation, the following sections briefly present the BLNS protocols, the relevant data, and the different functions considered in the implementation, according to previous Figure 8.2.

## C.1. Notation

- $[N] := \{1, 2, \ldots, N\}$.
- $\mathbb{Z}[X]$ is the ring of polynomials with coefficients in $\mathbb{Z}$.
- $\mathbb{Z}_q[X]$ is the ring of polynomials with coefficients in $\mathbb{Z}_q$.
- $\mathcal{R} := \mathbb{Z}[X]/(X^d + 1)$ is the ring whose elements are polynomials of the form $f = f_0 + f_1X + \cdots + f_{d-1}X^{d-1}$ with $f_0, f_1, \ldots, f_{d-1} \in \mathbb{Z}$, where addition is performed coefficient-wise, and multiplication is performed modulo $X^d + 1$.
- $\mathcal{R}_q := \mathbb{Z}_q[X]/(X^d + 1)$ is the ring whose elements are polynomials of the form $f = f_0 + f_1X + \cdots + f_{d-1}X^{d-1}$ with $f_0, f_1, \ldots, f_{d-1} \in \mathbb{Z}_q$, where addition is performed coefficient-wise modulo $q$, and multiplication is performed modulo $X^d + 1$.

## C.2. BLNS Protocols

To explain with a bit more of detail the structure of the BLNS framework, Figure C.1 shows the protocol between Holder and Verifier, for the issuing of a VC, while Figure C.2 shows the protocol between the Holder and the Verifier, for the verification of a VP.

$$
\begin{array}{lcl}
\mathsf{Holder}(\mathsf{ipk}, \mathsf{attrs}, \mathsf{idx}) & & \mathsf{Issuer}(\mathsf{isk}) \\
\mathsf{VC} \leftarrow \mathsf{emptyVC}() & & \\
\mathsf{VC.cp} \leftarrow \text{“BLNS23”} & & \\
\mathsf{VC.ipk} \leftarrow \mathsf{ipk} & & \\
\mathsf{VC.attrs} \leftarrow \mathsf{attrs} & & \\
(\rho_1, \mathsf{state}) \leftarrow \mathsf{Holder.VerCred1}(\mathsf{ipk}, \mathsf{attrs}, \mathsf{idx}) & & \\
\mathsf{attrs}' \leftarrow \mathsf{attrs}_{\mathsf{idx}} & & \\
 & \xrightarrow{\rho_1, \mathsf{attrs}', \mathsf{idx}} & \\
 & & \rho_2 \leftarrow \mathsf{Issuer.VerCred}(\mathsf{isk}, \mathsf{attrs}', \mathsf{idx}, \rho_1) \\
 & \xleftarrow{\rho_2} & \\
\mathsf{VC.pm} \leftarrow \mathsf{Holder.VerCred2}(\mathsf{ipk}, \rho_2, \mathsf{state}) & & \\
\mathbf{return}\ \mathsf{VC} & &
\end{array}
$$

**Figure C.1:** BLNS Issuing protocol

```
Holder(VC, idx)                                   Verifier(isk)
VP ← Holder.VerPres(VC, idx)
                          VP
                     ──────────→
                                                  if Verifier.Verify(VP) then
                                                      return accept
                                                  else
                                                      return reject
```

**Figure C.2:** BLNS Verification protocol

## C.3. Data

- ipk: public key of the issuer.
- isk: secret key of the issuer.
- attrs: set of attributes $(\mathsf{a}_1, \ldots, \mathsf{a}_l)$, where each $\mathsf{a}_i \in \{0,1\}^*$ is a string of bits of arbitrary length.
- idx: set of disclosed indices $(i_1, \ldots, i_k) \in \mathbb{Z}^k$.
- attrs′: set of disclosed attributes $(\mathsf{a}'_1, \ldots, \mathsf{a}'_k)$.
- cred: credential issued from the Issuer to the Holder. It is represented by a triple of elements $(\boldsymbol{s}, \boldsymbol{r}, x)$, where $\boldsymbol{s} \in \mathbb{Z}^{2d}$ is the short vector, $\boldsymbol{r} \in \mathcal{R}_q^{\ell_r}$ is sampled from a certain distribution and $x$ is generated uniformly random from the set $[N]$.

## C.4. Issuer's Functions

The Issuer's functions are the following:

- Issuer.KeyGen: This function takes no input and returns as output a pair (ipk, isk), where ipk is the public key of the issuer and isk is the secret key of the issuer.
- Issuer.VerCred: This function takes as input isk, the disclosed attributes attrs′, the set of disclosed indices idx, and the vector $\rho_1 = (u, \pi)$ which contains the commitment $u$ and the proof $\pi$ . It returns as output $\rho_2 = (\boldsymbol{s}, x)$, where $\boldsymbol{s}$ is the output of the GSampler algorithm, and $x$ is an integer uniformly sampled from the set $[N]$.

## C.5. Holders's Functions

The Holders's functions are the following:

- Holder.VerCred1: This function takes as input ipk, the attributes attrs and the set of disclosed indices idx. It returns as output $\rho_1$ and `state` which contains the polynomial vectors $\boldsymbol{m}$ and $\boldsymbol{r}$.
- Holder.VerCred2: This function takes as input ipk, $\rho_2$ and `state`. It returns as output the triple cred $= (\boldsymbol{s}, \boldsymbol{r}, x)$.
- Holder.VerPres: This function takes as input idx and the Verifiable Credential VC which includes some information such as ipk, attrs and cred. It returns as output a Verifiable Presentation VP, which includes some information such as ipk, attrs′, idx and the proof $\pi$.

*This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].*

## C.6. Verifier's Functions

The Verifier's functions are the following:

- Verifier.Verify: This function takes as input the Verifiable Presentation VP and returns $1$ if $\pi$ is valid, $0$ if the proof is invalid and $\perp$ otherwise.

## C.7. Other Functions

- NTRU.TrapGen: This function takes no input and returns the polynomial $a_1$ and a basis $\boldsymbol{B}$ of the $2d$-dimensional lattice $\Lambda$.
- GSampler: This function takes as input a polynomial $h \in \mathcal{R}_q$, a polynomial vector $\boldsymbol{a} \in \mathcal{R}_q^m$, basis $\boldsymbol{B} \in \mathbb{Z}^{2d \times 2d}$, standard deviation $\mathfrak{s} > 0$ and a center $\boldsymbol{c} \in \mathbb{Z}^d$. It returns as output the short vector $\boldsymbol{s}$ and $\boldsymbol{w} \in \mathcal{R}^m$.
- $\mathsf{Prove}_{\mathsf{Com}}^{\mathsf{H_{Com}}}$: This function takes as input a common reference string $\mathsf{crs}_{\mathsf{Com}}$, a statement x and a witness w. If all the checks pass, it returns the proof $\pi$.
- $\mathsf{Verify}_{\mathsf{Com}}^{\mathsf{H_{Com}}}$: This function takes as input a common reference string $\mathsf{crs}_{\mathsf{Com}}$, the proof $\pi$ and the statement x. It returns as output "accept" or "reject".
- $\mathsf{Prove}_{\mathsf{ISIS}}^{\mathsf{H_{ISIS}}}$: This function takes as input a common reference string $\mathsf{crs}_{\mathsf{ISIS}}$, a statement x and a witness w. If all the checks pass, it returns the proof $\pi$.
- $\mathsf{Verify}_{\mathsf{ISIS}}^{\mathsf{H_{ISIS}}}$: This function takes as input a common reference string $\mathsf{crs}_{\mathsf{ISIS}}$, a statement x and the proof $\pi$. It returns as output "accept" or "reject".
- $\mathsf{LHC.Com}_i$: This function takes as input $\boldsymbol{s}_i, \boldsymbol{y}_i \in \hat{\mathcal{R}}^{m_i}$ and a common reference string $\mathsf{crs}_i^{\mathsf{LHC}}$. It returns as output two values, a commitment com and st.
- $\mathsf{LHC.Open}_i$: This function takes as input a common reference string $\mathsf{crs}_i^{\mathsf{LHC}}$ and values $(\mathsf{com}, c)$ and st. If the check passes, it returns as output op.
- $\mathsf{LHC.Verify}_i$: This function takes as input a common reference string $\mathsf{crs}_i^{\mathsf{LHC}}$, $(\mathsf{com}, c)$ and two values $(\boldsymbol{z}, \mathsf{op})$. It returns as output "accept" or "reject".
- Rej: This function takes as input two vectors of the same length $\vec{z}$ and $\vec{x}$, and two scalars $s$ and $M$. It returns as output "reject" or "accept".

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

## D. Experimental Evaluation of PQ Hash Based Signature Algorithms

Table D.1 shows the comparison between LMS and SPHINCS$^+$ that has been done in [2], where the algorithms were run on an Intel® Xeon CPU @ 2.20 GHz with 2 cores and 7.68 GB RAM. Tests run 1000 times for each parameter set. For this table, signature generation and verification time values were converted from mean of Mcycles to ms. The used LMS implementation is the Cisco one [135], while the original SPHINCS$^+$ implementation [136] was used.

Table D.2 shows the performance comparison of LMS and XMSS done in [3], where [135] was used as LMS implementation, while XMSS was tested using the official implementation [137]. The algorithms were run on an ARM Cortex-M4 board, to test the feasibility of hash-based PQ algorithms on embedded systems. For signature generation and verification, the values were converted from mean of Mcycles to ms, assuming a 200 MHz frequency.

**Table D.1:** Performance of LMS and SPHINCS$^+$ algorithms [2].

| Algorithm | NIST Security Level | Public Key Size (Bytes) | Sign Size (KBytes) | Keypair Generation (ms) | Sign Generation (ms) | Sign Verification (ms) |
|---|---|---|---|---|---|---|
| LMS-SHA2-256 h=15, w=4 | 5 | 60 | 2,67 | 2519 | 0,520 | 0,168 |
| LMS-SHA2-256 h=15, w=8 | 5 | 60 | 1,62 | 13720 | 2,835 | 1,298 |
| LMS-SHA2-256 h=20, w=4 | 5 | 60 | 2,83 | 3222 | 0,666 | 0,170 |
| LMS-SHA2-256 h=20, w=8 | 5 | 60 | 1,78 | 19373 | 4,003 | 1,299 |
| SLH-DSA-SHA2-256 h=15, w=16 | 5 | 64 | 17,28 | 3 | 80,022 | 0,464 |
| SLH-DSA-SHA2-256 h=15, w=256 | 5 | 64 | 14,11 | 26 | 151,900 | 3,202 |
| SLH-DSA-SHA2-256 h=20, w=16 | 5 | 64 | 19,58 | 3 | 83,384 | 0,628 |
| SLH-DSA-SHA2-256 h=20, w=256 | 5 | 64 | 15,36 | 26 | 178,888 | 4,633 |

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

**Table D.2:** Performance comparison of LMS and XMSS on ARM Cortex-M4 using SHA-256 as hash function [3].

| Algorithm | NIST Security Level | Public Key Size (Bytes) | Sign Size (KBytes) | Keypair Generation (ms) | Sign Generation (ms) | Sign Verification (ms) |
|---|---|---|---|---|---|---|
| LMS-SHA2-256 h=5, w=16, d=1 | 5 | 60 | 2,352 | 589,945 | 632,584 | 12,883 |
| LMS-SHA2-256 h=10, w=16, d=1 | 5 | 60 | 2,512 | 18874,411 | 18955,790 | 13,294 |
| XMSS-SHA2-256 h=5, w=16, d=1 | 5 | 68 | 2,340 | 1216,273 | 1238,631 | 16,037 |
| XMSS-SHA2-256 h=10, w=16, d=1 | 5 | 68 | 2,500 | 38922,540 | 38942,822 | 18,382 |

This project has received funding from the European Union under the Horizon Europe framework programme [grant agreement no. 101119746].

Quantum-oriented Update to Browsers and Infrastructures for the PQ transition (QUBIP)

https://www.qubip.eu

D1.4 – Analysis and design of PQ building blocks

Version 1.0

Horizon Europe