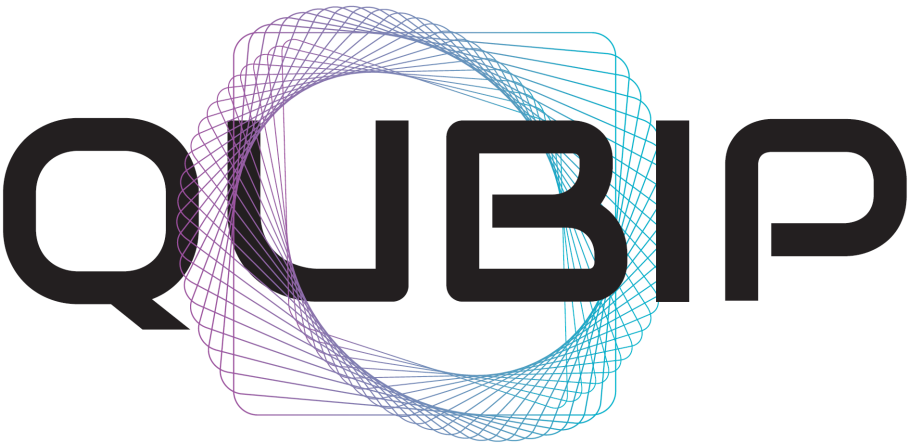
Horizon Europe



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

Specifications of Pilot Demonstrators

Deliverable number: D2.1

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:	Javier Faba – UPM Juan Pedro Brito – UPM
Deliverable nature:	Report (R)
Dissemination level:	Public (PU)
Contractual Delivery Date:	31 October 2024
Actual Delivery Date:	29 October 2024
Number of pages:	43
Keywords:	cybersecurity, post-quantum cryptography, pilot demonstrators.
Contributors:	Andrea Vesco – LINKS Davide Margaria – LINKS Antonio Lioy – POLITO Grazia D'Onghia – POLITO Silvia Sisinni – POLITO Davide Bellizia – TELSY Alberto Battistello – SECPAT Eros Camacho-Ruiz – CSIC Piedad Brox – CSIC Antonio Pastor – TID Luis F. Gonzalez – UC3M Nicola Tuveri – TAU Akif Mehmood – TAU Daniel Luoma – TAU Alex Shaindlin – TAU Nouman Ali Khan – TAU Dmitry Belyavskiy – REDHAT Federico Parente – SMART Andrés del Álamo – CIB
Peer review:	Antonio Lioy – POLITO Davide Bellizia – TELSY
Approved by:	ALL partners

Table 1: Document revision history

Issue Date	Version	Comments
24/04/2024	0.1	Initial table of contents.
30/09/2024	0.2	First draft for internal review.
29/10/2024	1.0	Final version for submission.

Abstract

This document is the Deliverable D2.1 of the Quantum-oriented Update to Browsers and Infrastructures for the Post-quantum transition (QUBIP) project. The document describes three pilot demonstrators together with their system requirements and specifications and the overall system architecture. Note that the three pilots are based on the eight Post-Quantum (PQ) building blocks presented in Deliverable D1.4.

Contents

List of Figures

List of Tables

List of Acronyms

AK	Attestation Key
API	Application Programming Interface
BBS	Boneh, Boyen, and Shacham
BLNS	Bootle, Lyubashevsky, Nguyen, and Sorniotti
CA	Certification Authority
CCIPS	Centrally Controlled IPSec
CNF	Container Network Function
COTS	Commercial Off-the-Shelf
CRQC	Cryptographically Relevant Quantum Computer
CSV	Comma-Separated Values
DID	Decentralized IDentifier
ECDSA	Elliptic Curve Digital Signature Algorithm
EdDSA	Edwards-curve Digital Signature Algorithm
EK	Endorsement Key
ESP	Encapsulating Security Payload
EU	European Union
FAQ	Frequently Asked Question
FESCo	Fedora Engineering Steering Committee
fTPM	firmware Trusted Platform Module
GDPR	General Data Protection Regulation
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol over Secure Socket Layer
HW	Hardware
I/O	Input/Output
I2NSF	Interface to Network Security Function
IKE	Internet Key Exchange
IMA	Integrity Measurement Architecture
IoT	Internet of Things
IP	Intellectual Property
IPsec	IP Security
JPT	JSON Proof Token
JSON	JavaScript Object Notation
JWT	JSON Web Token
K8s	Kubernetes
KEM	Key Encapsulation Method
L2S-M	Link-Layer Secure connectivity for Microservice platforms
LMS	Leighton-Micali Hash-Based Signature
MANO	Management and Orchestration
MCU	Micro-Controller Unit
ML-DSA	Module-Lattice-Based Digital Signature Algorithm
ML-KEM	Module-Lattice-Based Key-Encapsulation Mechanism
MPU	Micro-Processor Unit
MQTT	Message Queuing Telemetry Transport
NFV	Network Functions Virtualization

NIST	National Institute of Standards and Technology
NSS	Network Security Services
NSS	Network Security Services
OCSP	On-line Certificate Status Protocol
ONOS	Open Network Operating System
OP-TEE	Open Portable Trusted Execution Environment
OS	Operating System
PCR	Platform Configuration Register
PKCS	Public-Key Cryptography Standards
PKI	Public-Key Infrastructure
PKIX	Public-Key Infrastructure using X.509
PLC	Programmable Logic Controller
PQ/T	Post-Quantum/Traditional
PQ	Post-Quantum
PQC	Post-Quantum Cryptography
QKD	Quantum Key Distribution
QUBIP	Quantum-oriented Update to Browsers and Infrastructures for the Post-quantum transition
RA	Remote Attestation
RAM	Random Access Memory
RCA	Root Certification Authority
REST	REpresentational State Transfer
RoT	Root of Trust
RTR	Root of Trust for Reporting
RTS	Root of Trust for Storage
SDN	Software-Defined Networking
SE	Secure Element
SLH-DSA	Stateless Hash-Based Digital Signature Algorithm
SNI	Server Name Indication
SotA	State of the Art
SSI	Self-Sovereign Identity
TCG	Trusted Computing Group
TCP	Transmission Control Protocol
TEE	Trusted Execution Environment
TFS	TeraFlow SDN
TLS	Transport Layer Security
TPM	Trusted Platform Module
UI	User Interface
UX	User eXperience
VC	Verifiable Credential
VP	Verifiable Presentation
VXLAN	Virtual eXtensible Local Area Network
W3C	World Wide Web Consortium
ZK	Zero-Knowledge
ZKP	Zero-Knowledge Proof

1 Introduction

The QUBIP project focuses on the transition from traditional cryptography to Post-Quantum Cryptography (PQC) of protocols, network, and systems we use today.

The two main goals of the project are:

1. to simplify the transition process and make it replicable through recommended practices, structured support processes for industry and contributions to EU standardisation and policymaking;
2. to counter PQ threats as soon as possible.

The QUBIP project contributes to the definition of a replicable and reference transition process by maximising the return on experience from three practical transition exercises. The exercises involve the tailored adoption of eight PQ building blocks into three main systems. These PQ building blocks, which transition to PQC as been addressed in Deliverable D1.4, will be properly integrated into three pilot demonstrators in relevant use cases to address system-level transition challenges. This document describes the system specifications and requirements and the overall architecture of those three pilot demonstrators.

Before properly describing each pilot demonstrator and understanding the reasons behind their evaluation, it is of value to remark the methodology followed by QUBIP, which is depicted in Figure 1.1.

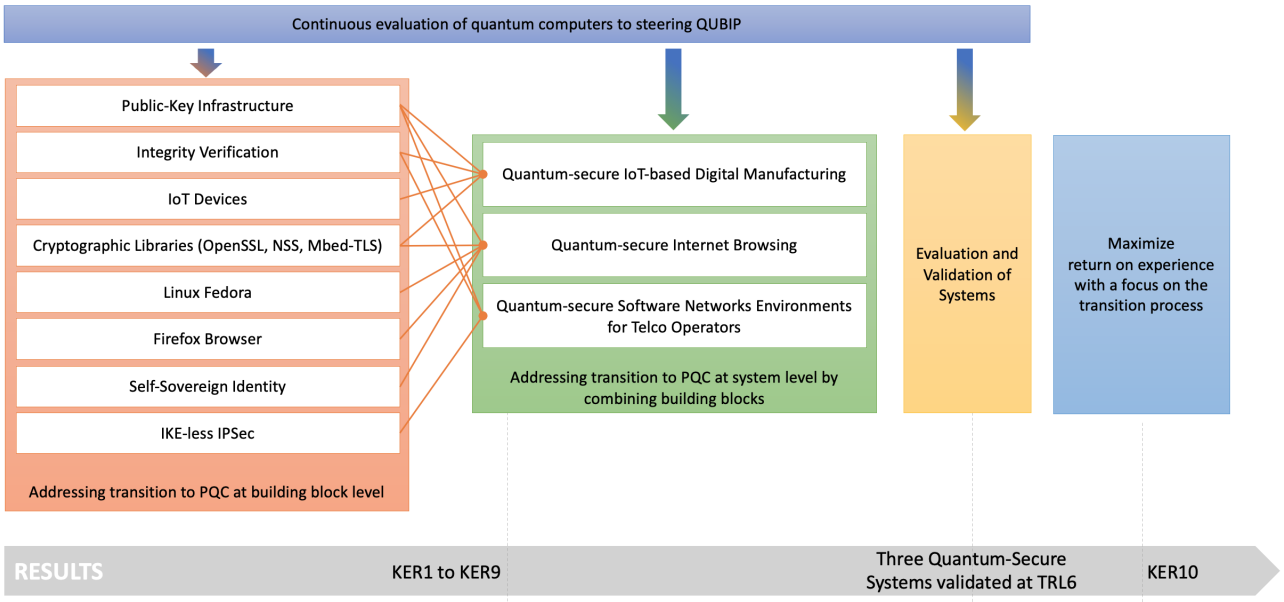


Figure 1.1: Methodology of the QUBIP project: from building block to system level transition to PQC

The first and fundamental step is to continuously evaluate the State of the Art (SotA) of the Cryptographically Relevant Quantum Computers (CRQCs) and PQ algorithms to provide the partners relevant information when selecting PQC at different levels and stages. This assessment, presented in Deliverable D1.1 and subsequent updates, includes various technological implementations and today's relevant PQ algorithms. The feedback from this evaluation has supported and will support the transition of the eight building blocks to PQC presented in Deliverable D1.4.

Then, the eight building blocks will be integrated into three pilot demonstrators:

1. Quantum-secure Internet of Things (IoT)-based Digital Manufacturing.

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

2. Quantum-secure Internet Browsing.
3. Quantum-secure Software Networks Environments for Telco Operators.

The combination of these heterogeneous building blocks into the three pilot demonstrators is expected to raise a number of dependency issues, which will be resolved during the integration phase to generate further knowledge about the transition process to PQC. This experience, together with appropriate evaluation and validation activities of the systems, will be reflected in a replicable transition process to PQC.

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 [1].

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

2 Quantum-secure IoT-based Digital Manufacturing

The Quantum-secure IoT-based Digital Manufacturing pilot has been designed to test quantum-secure data exchange between different IoT devices that typically comprise a connected digital manufacturing system. In this pilot, both flavours of the same IoT device (i.e., MCU and MPU-based), are securely connected with a central Data Server whose mission is to store and process critical data collected from the field. The system is designed for:

- data acquisition from external sensors directly connected to IoT devices;
- quantum-secure communication between IoT devices and the Data Server at the transport layer;
- deployment of the Message Queuing Telemetry Transport (MQTT) protocol at application layer for IoT data exchange;
- software integrity verification by means of a Remote Attestation (RA) protocol of Micro-Processor Unit (MPU)-based IoT devices.

The pilot demonstrator also sets a Public-Key Infrastructure (PKI) with the following components:

- Certification Authority (CA): is a trusted entity responsible for issuing, managing, and revoking digital certificates. It verifies the identity of entities requesting certificates and ensures the authenticity of their public keys.
- Root Certification Authority (RCA): acts as an intermediary between the entities (i.e., IoT devices) and the CA. It verifies the identity of devices before the CA issues a digital certificate. The RCA does not issue certificates directly but plays a crucial role in the verification process.
- Digital Certificates: securely bind a public key with an entity in the system, e.g. IoT devices, Data Server.
- Public and Private Keys: are used by system components for encryption, decryption, and authentication.
- Secure Element (SE): integrated within the IoT devices, it securely stores the private keys of the MPU and Micro-Controller Unit (MCU), protecting them from unauthorized access. It also performs key operations without exposing the private key, such as signing authentication tokens or decrypting incoming messages.
- On-line Certificate Status Protocol (OCSP) responder: used during Transport Layer Security (TLS) handshake by system entities to verify the validity of a certificate in real time.
- Attestation Agent: it uses the device's private key, stored in the PQ firmware Trusted Platform Module (fTPM), to sign an integrity report when requested by a Remote Verifier. The Remote Verifier uses the device's public key certificate to verify the authenticity of the report before evaluating it.

2.1 Requirements

Table 2.1 shows the requirements of the system to fulfill the objectives described above. In that sense, the requirements can be organized as it follows:

- Req-DM#01 and Req-DM#02 are related to PKI management in terms of digital certificates and OCSP verification.
- Req-DM#03 is related to the SE.

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

- Req-DM#04, Req-DM#05, Req-DM#06 and Req-DM#07 are the specific requirements of remote attestation, highlighting the importance to verify whether MPU-based IoT devices can be trusted or not through integrity verification techniques.
- Req-DM#08 and Req-DM#09 are related to data collection and exchange.
- Req-DM#10 defines the need to establish TLS channels that support quantum-secure communication between clients and servers.
- Req-DM#11 defines the need for an application protocol that is crucial for communication between software applications.
- Req-DM#12 highlights the importance of adopting different cryptographic algorithms at transport and application layers.
- Req-DM#13 defines the need for interoperability between MbedTLS and OpenSSL TLS implementations to ensure MCU and MPU-based IoT devices are interchangeable.
- Req-DM#14 highlights the importance of having a complete system documentation.

Table 2.1: IoT-based Digital Manufacturing Pilot requirements

Req.ID	Name	Description
Req-DM#01	Certificate Authority	The CA MUST be able to generate and sign certificates that use PQC.
Req-DM#02	OCSP Verification	Digital certificates used in TLS protocol MUST be verified (by means of validity) by the TLS client through OCSP.
Req-DM#03	Secure Element	The SE MUST store the private key of the IoT device.
Req-DM#04	Integrity Verification	The MPU-based IoT devices MUST be able to report on their status of integrity.
Req-DM#05	Remote Verifier	The Remote Verifier MUST have PQC support in order to check the authenticity of the report sent by the Attestation Agent.
Req-DM#06	PQ fTPM	The PQ fTPM running inside the Trusted Execution Environment (TEE) MUST store the private part of the PQ key used to sign the `quote`.
Req-DM#07	Attestation Agent	The Attestation Agent MUST be able to get the PQ integrity report and forward it to the Remote Verifier.
Req-DM#08	Data Collection	The IoT device MUST be able to collect data from external sensors (e.g., manufacturing, energy consumption, movement).
Req-DM#09	Data Exchange	The IoT device and the Data Server MUST be able to exchange data through secure channels.
Req-DM#10	Quantum-secure TLS Channel	Quantum-secure TLS channel MUST be used between clients and the server.
Req-DM#11	Application	Clients and the Data Server MUST exchange data through an application protocol suited for IoT devices.
Req-DM#12	Cryptographic Algorithms	The system MUST support Post-Quantum/Traditional (PQ/T) hybrid implementations of key exchange and digital signatures, and quantum-secure encryption/decryption algorithms.

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

Req.ID	Name	Description
Req-DM#13	Interoperability	MbedTLS-based TLS clients MUST be compatible with OpenSSL-based TLS servers.
Req-DM#14	Documentation	Documentation covering the steps to setup and run all instances.

2.2 Specifications

Table 2.2 shows the system specifications, based on the Table 2.1 of requirements.

Table 2.2: IoT-based Digital Manufacturing Pilot specifications

Spec.ID	Name	Description	Req.ID
Spec-DM#01	Supported Certificates	The pilot supports PQ/T hybrid X.509 certificates.	Req-DM#01
Spec-DM#02	Composite Certificates	Digital certificates support a hybrid cryptographic scheme, incorporating both classical cryptographic signatures (e.g., ECDSA) and PQC signatures within a composite certificate structure.	Req-DM#01
Spec-DM#03	Multifactor Authentication	The RCA uses multifactor authentication mechanisms for device registration, such as public-key-based challenge-response protocols and secure communication channels (i.e., TLS).	Req-DM#01
Spec-DM#04	Traceability	The RCA system logs registration attempts, ensuring traceability of device identity and registration processes.	Req-DM#01
Spec-DM#05	Cryptographic Binding	Device registration includes cryptographic binding of the device identity with the registration authority, ensuring tamper resistance.	Req-DM#01
Spec-DM#06	OCSP	The PKI implements an OCSP responder to provide real-time certificate validation.	Req-DM#02
Spec-DM#07	SE Storage	The SE supports secure key generation, storage, and usage, ensuring the private key never leaves the SE unencrypted.	Req-DM#03
Spec-DM#08	Integrity Measurement	The MPU-based IoT devices include an integrity measurement mechanism that regularly checks the state of system components (e.g., firmware, software, configuration files).	Req-DM#04
Spec-DM#09	Integrity Storage	Any integrity measurement is stored in the PQ fTPM, protected from unauthorized modifications.	Req-DM#04
Spec-DM#10	PQ Secure Boot	Each modifiable boot partition is authenticated with Leighton-Micali Hash-Based Signature (LMS) PQ algorithm by the subsequent one before being loaded.	Req-DM#04

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

Spec.ID	Name	Description	Req.ID
Spec-DM#11	Remote Verifier Location	The Remote Verifier has direct communication with the Attestation Agent.	Req-DM#05
Spec-DM#12	PQ Measured Boot	The PQ fTPM creates integrity reports signed with hash-based PQ algorithms.	Req-DM#06
Spec-DM#13	Attestation Agent	The Attestation Agent requests generation of PQ-quotes to the PQ fTPM and forwards them together with IMA measurement log to the Remote Verifier.	Req-DM#07
Spec-DM#14	PLC Connection	Both MPU-based and MCU-based IoT devices are connected to Programmable Logic Controllers (PLCs) through ModBus/Transmission Control Protocol (TCP).	Req-DM#08
Spec-DM#15	MCU Data Acquisition	100 byte payload.	Req-DM#08
Spec-DM#16	MPU Data Acquisition	200 byte payload.	Req-DM#08
Spec-DM#17	IoT Device with Hybrid TLS	IoT devices uses TLS 1.3 with both classical and PQ algorithms to establish secure channels.	Req-DM#09
Spec-DM#18	Server Data Storage	The server provides data storage, either internally or on an external data server.	Req-DM#09
Spec-DM#19	Client Definition	IoT devices only act as clients at both transport and application layers.	Req-DM#10
Spec-DM#20	Client Communication Protocol	The client uses a PQ/T hybrid TLS 1.3 to establish secure communication channels.	Req-DM#10
Spec-DM#21	Server Communication Protocol	The server uses PQ/T hybrid TLS 1.3 to establish secure communication channels.	Req-DM#10
Spec-DM#22	Application Protocol	Clients and the server exchange application data through the MQTT protocol.	Req-DM#11
Spec-DM#23	MQTT Client – MPU	The MQTT client supports OpenSSL integration.	Req-DM#11
Spec-DM#24	MQTT Client – MCU	The MQTT client supports MbedTLS integration.	Req-DM#11
Spec-DM#25	MQTT Broker OpenSSL support	The MQTT Broker supports OpenSSL 3.2.1.	Req-DM#11
Spec-DM#26	MQTT Broker	The MQTT Broker is based on Mosquitto broker.	Req-DM#11
Spec-DM#27	MQTT Data Transmission – MCU	1 MQTT message per second.	Req-DM#11
Spec-DM#28	MQTT Data Transmission – MPU	5 MQTT messages per second.	Req-DM#11

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

Spec.ID	Name	Description	Req.ID
Spec-DM#29	KEM Algorithms	The hybrid implementation of Key Encapsulation Method (KEM) algorithms uses X25519 and ML-KEM.	Req-DM#12
Spec-DM#30	Signature Algorithms	The hybrid implementation of signature algorithms uses EdDSA, ML-DSA and SLH-DSA.	Req-DM#12
Spec-DM#31	Cipher Algorithms	AES-128, AES-256.	Req-DM#12
Spec-DM#32	Hash Algorithms	SHA-2, SHA-3, and their derivatives.	Req-DM#12
Spec-DM#33	Interoperability	MbedTLS-based and openSSL-based TLS endpoints uses the same PQ/T hybrid algorithms for full interoperability.	Req-DM#13
Spec-DM#34	Client Documentation	Documentation covering the steps to setup and run all instances of client applications and connect to the server.	Req-DM#14
Spec-DM#35	Server Documentation	Documentation covering the steps to setup and run all instances at server side.	Req-DM#14

2.3 Architecture

The architecture of the IoT-based Digital Manufacturing pilot is depicted in Figure 2.1. This pilot relies on connected IoT devices, which come in two flavours: MPU-based and MCU-based. Their designs are detailed in Deliverable D1.4.

Apart from the differences in the HW platform, the main differences between the two flavours concern the cryptographic library used and the internal communication interface with the SE:

- **MPU-based devices**: use the OpenSSL library and communicate via a shared memory-mapped interface with the SE.
- **MCU-based devices**: use the MbedTLS cryptographic library and communicate via a serial interface connection with the SE.

The IoT device, acting as a client, establishes a quantum-secure TLS channel with the MQTT Broker, acting as a server. In that sense, the MQTT Broker acts as a central hub for managing messages between devices in the MQTT-based communication system. The system uses a publish/subscribe model, in which the publisher (i.e., IoT devices) publishes messages to specific topics and the subscriber (i.e., the Data Server) consumes the published messages. Both flavours of IoT devices collect data from specific sensors through their Input/Output (I/O) interfaces. Once the quantum-secure communication channel is established, the IoT devices send the collected data to the MQTT Broker, which in turn publishes data on specific topics. The Data Server subscribes to the respective topics and consumes data.

In addition, a RA process is deployed to continuously monitor the health status of MPU-based IoT devices. RA relies on the presence of a Remote Verifier, deployed on the same node as the MQTT broker or on another node, that challenges IoT devices to prove their trustworthy status through a tamper-proof integrity evidence. If all MPU-based IoT devices are assessed as trustworthy, the system continues to operate as described above. If instead the Remote Verifier detects signs of compromise on one or more MPU-based IoT devices, it promptly notifies the MQTT Broker that a device, with a particular device ID, has been compromised. From that moment on, the MQTT Broker isolates the compromised device by refusing connections from it (and closing any on-going connections), to prevent the injection into the Data Server of data coming from tampered devices (i.e., the injection of potentially malicious data).

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

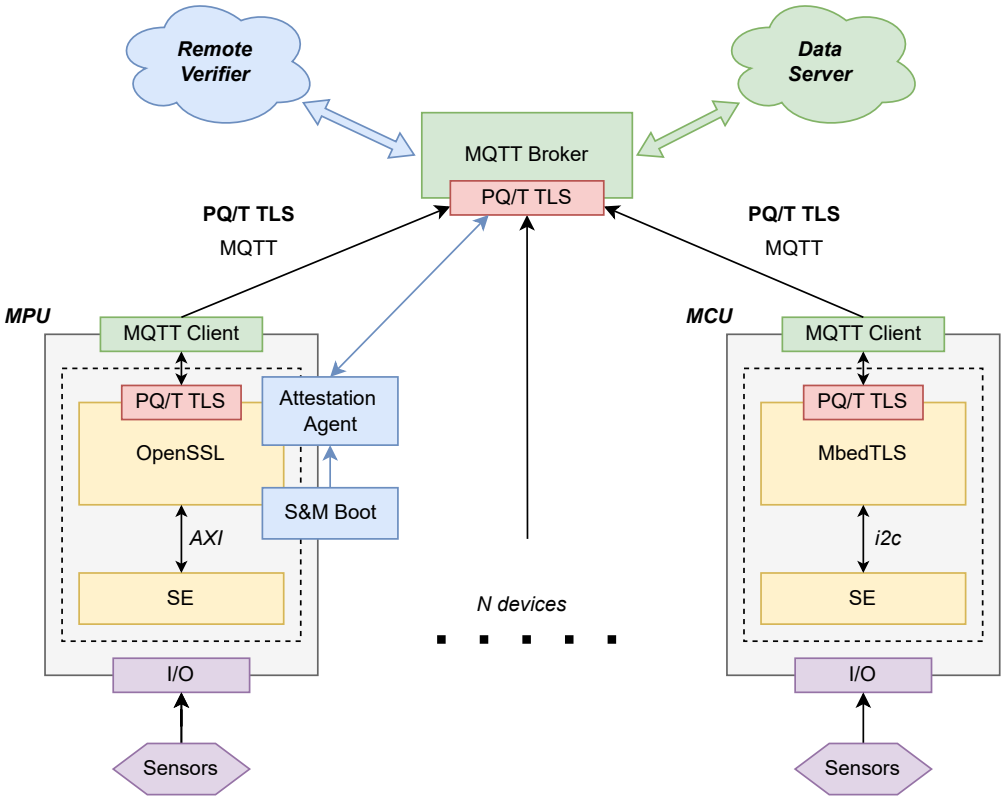


Figure 2.1: Architecture of the IoT-based Digital Manufacturing Pilot.

2.3.1 Sequence of interactions

Figure 2.2 shows the sequence diagram of interactions among components involved in the data exchange. The IoT devices establish a quantum-secure communication channel with the MQTT Broker using TLS 1.3. A quantum-secure communication channel can also be established between the MQTT Broker and the Data Server.

At the application layer, the MQTT protocol rules the data exchange between IoT devices, MQTT Broker, and Data Server, in accordance with the publish/subscribe model. The system works as it follows:

- Client Authentication: the MQTT client (i.e., IoT device) initiates a connection with the MQTT Broker by providing a device ID, username, and password. Data is sent over the established TLS channel. The MQTT Broker checks the provided credentials to authenticate the MQTT client.
- Authorization: Once authenticated, the MQTT Broker verifies what permissions the MQTT client has, whether it can publish data, subscribe to topics, or perform both actions. This step ensures that MQTT clients are only granted the level of access they need.
- Connection Acknowledgment: if authentication and authorization are successful, the MQTT Broker sends an acknowledgment (ACK) signal to the MQTT client, indicating that the connection has been established successfully.
- Subscription: The Data Server sends a subscription request to the MQTT Broker. Then, the MQTT Broker internally adds this subscription and sends an ACK signal back to the Data Server.
- Data Publishing: The MQTT client can then publish the data to specific topics. The MQTT Broker, acting as an intermediary, forwards the published data to all subscribers (i.e., the Data Server). This ensures that the data reaches the intended destination, typically for storage or further processing.

Along with the application logic, the system supports procedures to ensure that MPU-based IoT devices,

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

participating in the data collection and exchange, are trustworthy. This flavour of IoT devices can boot only with a authentic and healthy system images, as a PQ Secure Boot is implemented to verify that the system components, loaded at startup, come from an authorized provider and have not been tampered with. The verification of authenticity of system components during boot is performed using PQ algorithms, as described in the Deliverable D1.4. Furthermore, a PQ RA protocol is implemented and integrated into the system to allow a Remote Verifier to check that an IoT device has performed a trusted boot, and that it remains in a trustworthy status during its operation. To support this verification, a Measured Boot process is implemented in the device firmware, meaning that a firmware component acquires the integrity measure of the next firmware component after having loaded it into memory and before passing control of the device to it. The integrity measurements collected during the Measured Boot are recorded in a data structure compliant with Trusted Computing Group (TCG) Event Log [2], and stored in a secure memory area. MPU-based IoT devices use a TEE, specifically Open Portable Trusted Execution Environment (OP-TEE), to run a PQ fTPM. The PQ fTPM initializes the Platform Configuration Registers (PCRs), related to the boot components (typically PCRs 0-9), with the corresponding measurements contained in the TCG Event Log, acquired during Measured Boot. The chain of measurements acquired during Measured Boot is also extended to the runtime of the device, by relying on the Integrity Measurement Architecture (IMA) module provided by the Linux kernel. This module, enabled by appropriate policies, performs measurements on all software components loaded during runtime and matching the configured IMA-policies. The fTPM driver exposes the PQ fTPM running inside OP-TEE as a physical Trusted Platform Module (TPM) device that can be used by other kernel modules and user space libraries and applications. The IMA module relies on this fTPM driver to extend the integrity measurements, acquired during runtime, into a specific PCR (typically PCR 10) of the PQ fTPM. The use of PQ fTPM and integrity measurements allows boot and run-time trustworthiness to be verified by the Remote Verifier through a RA protocol.

Figure 2.3 shows the sequence diagram of the RA protocol for integrity verification. The Attestation Agent in each MPU-based IoT device starts immediately after the initialization of the Linux user-space. Once started, the Attestation Agent requests the PQ fTPM to generate an Attestation Key (AK), indicating a PQ cryptographic algorithm among those supported by the PQ fTPM, as described in the Deliverable D1.4. Then, it executes a protocol based on the `activate credential` command (supported by the TPM specification), to securely register in the Remote Verifier the AK and the Endorsement Key (EK), which identifies the PQ fTPM. At this point, the Attestation Agent waits to receive attestation challenges from the Remote Verifier. An attestation challenge is an HTTP request containing a nonce, i.e., a statistically unique random number needed to protect the protocol from replay attacks, the list of PCRs that the Remote Verifier wishes to check, and the PCR bank, which indicates the hash algorithm used to extend the integrity measurements. Upon receiving the attestation challenge, the Attestation Agent requests the PQ fTPM to create a `quote`, indicating the nonce, the list of PCRs, their relative bank, and the handle to the PQ AK previously generated. The PQ fTPM creates the `quote`, consisting of a PQ signature on the content of the requested PCRs concatenated to the nonce, by using the secret part of the PQ AK. The Attestation Agent then creates the integrity report, containing the `quote` generated by the PQ fTPM, the values of PCRs signed in the `quote`, and the Measurement Log generated by IMA (whose integrity is protected and verifiable through one of the PCRs), and sends it to the Remote Verifier. The Remote Verifier then performs all the checks to verify that the information contained in the received integrity report is fresh, authentic, and representative of a trustworthy device. In case an integrity failure is detected, the Remote Verifier notifies the MQTT Broker to isolate the compromised IoT device. All communication between the Attestation Agent and the Remote Verifier, and possibly between the Remote Verifier and the MQTT Broker, is protected by a quantum-secure TLS channel.

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

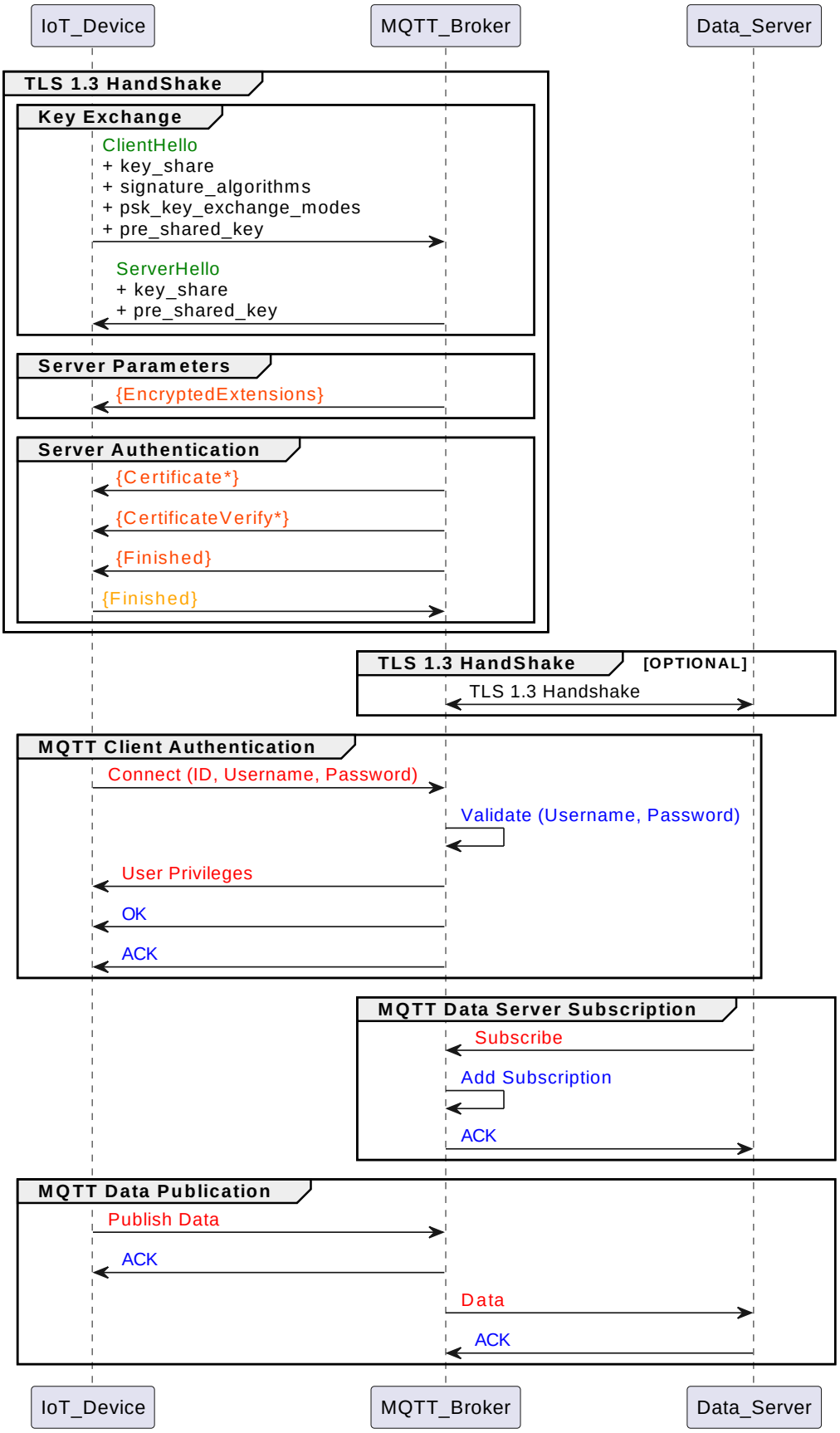


Figure 2.2: Sequence diagram of the interactions between MQTT client, MQTT Broker and Data Server

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

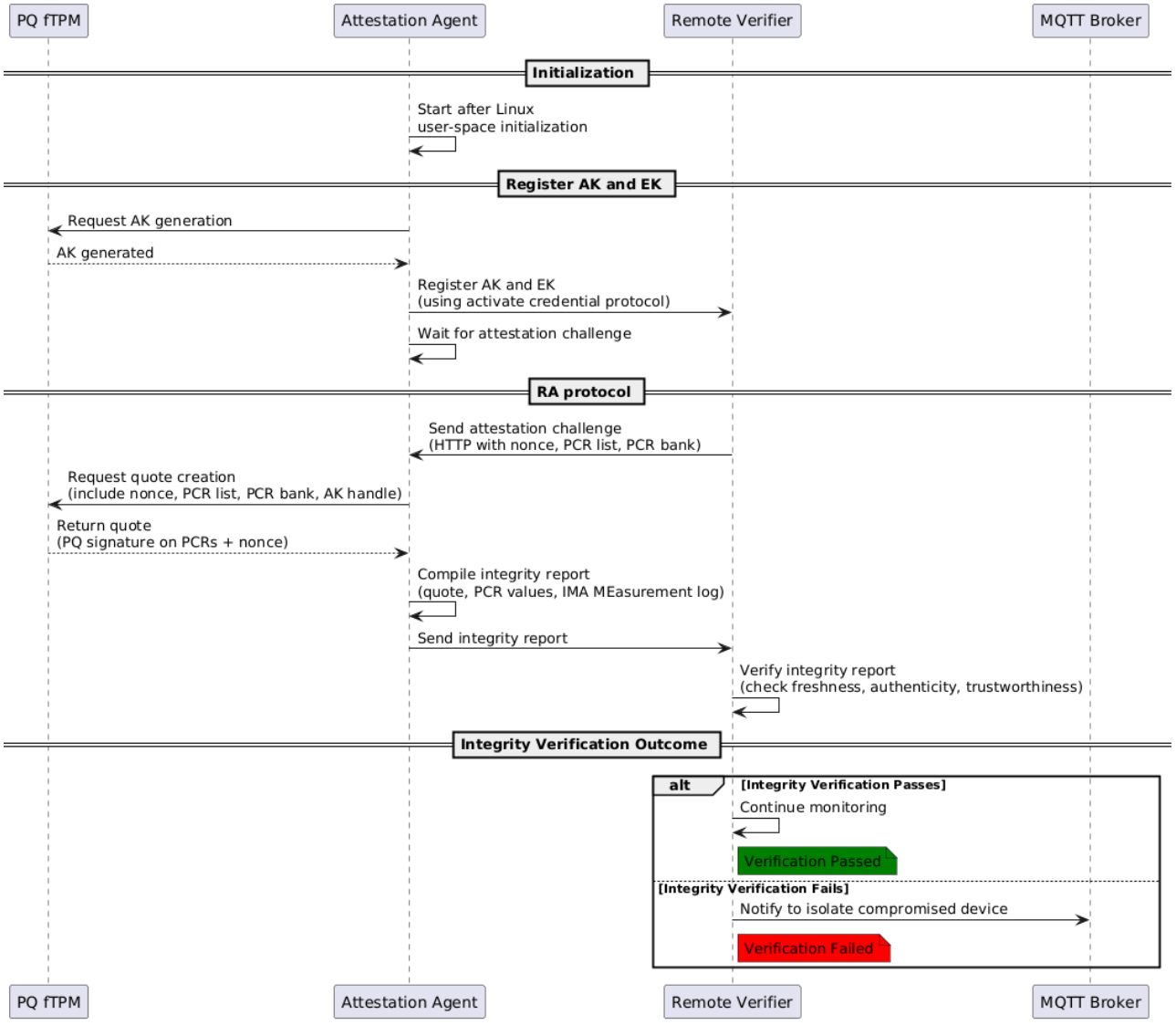


Figure 2.3: Sequence diagram of the remote attestation process of MPU-based IoT devices.

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

3 Quantum-secure Internet Browsing

The Quantum-secure Internet Browsing pilot is designed to explore a secure way to browse the Internet by integrating advanced cryptographic methods to protect data and user identities against the additional threats posed by the advent of CRQCs.

Key components of the pilot include OpenSSL and Network Security Services (NSS), which provide the applications abstractions to handle encryption and secure communications. OpenSSL, with the QUBIP provider, enables the use of PQ/T Hybrid algorithms for key exchange and authentication. The NSS libraries often empower applications facing users on their client devices. In this pilot, in particular, NSS lays the foundations for managing cryptographic operations within the Mozilla Firefox Internet browser, ensuring secure interactions.

Overall, this pilot seeks to achieve the following goals:

- secure and privacy-preserving Internet browsing using PQ/T Hybrid methods;
- validation of PQC-enabled Decentralized IDentifiers (DIDs) and Verifiable Credentials (VCs) in a web environment;
- testing realistic deployments of PQ/T Hybrid algorithms in existing applications through OpenSSL and NSS.

Client components. The Client host includes several key building blocks, each serving a distinct function. IOTA Identity provides a decentralized identity management framework. Mozilla Firefox, an open-source web browser renowned for its security and privacy features, relies on NSS to handle critical cryptographic operations and secure communication protocols. NSS is a comprehensive library suite, originally developed by Netscape, that supports cross-platform development of security-enabled applications, particularly for TLS, PKI, and cryptographic algorithms. The QUBIP `Public-Key Cryptography Standards (PKCS) #11` Module interfaces with an external software library that implements PQC algorithms, enabling TLS 1.3 support for PQ/T Hybrid key exchange and authentication (i.e., the Client application can leverage PQ/T Hybrid PKIX certificates to ensure server-side authentication). Notably, QUBIP does not create new PQC implementations, but rather allows the selection of arbitrary external implementations that meet with users' specific performance and security requirements. Finally, Fedora Linux serves as the operating system, providing a cutting-edge, secure, and flexible environment.

Server components. On the Server host, the building blocks include the Nginx web server, a high-performance, open-source web and reverse proxy server recognized for its stability, low resource usage, and scalability. The Verifier component is responsible for validating and authenticating digital signatures or certificates, ensuring data integrity and authenticity. OpenSSL is a comprehensive, open-source toolkit that implements TLS and other protocols while offering an extensive cryptography library. The QUBIP Provider functions as the interface layer between OpenSSL and the external implementations, facilitating the integration of advanced PQ cryptographic functions. The QUBIP Provider ultimately enables the OpenSSL stack to establish TLS 1.3 connections supporting PQ/T Hybrid key exchange and authentication. In particular, OpenSSL-based programs can leverage the capabilities of the QUBIP Provider to authenticate and verify PQ/T Hybrid PKIX certificate chains. The PQC External Software/Hardware Implementation combines software and/or specialized hardware to deploy PQC algorithms, enhancing security against quantum threats. It is important to note that the implementations selected for the QUBIP NSS module can be different from the ones chosen for the QUBIP OpenSSL Provider: as long as client and server share

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

a common set of PQ/T Hybrid algorithms, the server use case and client use case could pick different implementations for the same algorithm, granting an extra degree of freedom in selecting the performance and security requirements that best suit different usage scenarios. Like on the Client host, Fedora Linux serves as the operating system, providing a reliable platform for managing server-side applications.

3.1 Requirements

Table 3.1 shows the requirements of the pilot demonstrator. The requirements can be roughly divided into three groups: internal technical requirements, deployment infrastructure requirements, and sociotechnical requirements (those related to "human factors" such as documentation, education, and user feedback). Req-IB#01 through Req-IB#08 are internal technical requirements; Req-IB#09 through Req-IB#14, as well as Req-IB#17 and Req-IB#18, are deployment infrastructure requirements; Req-IB#15, Req-IB#16, Req-IB#19, and Req-IB#20 are sociotechnical requirements. Req-IB#21 through Req-IB#23 bridge the latter two categories, concerning the user-facing aspects of how the demonstrator is deployed and tested.

Table 3.1: Internet Browsing Pilot requirements

Req.ID	Name	Description
Req-IB#01	Issuer	Service for the User to request, update, and revoke a VC.
Req-IB#02	Digital Wallet	A wallet implemented as a browser extension to handle VCs, DIDs and key material, and to generate VP (in plain and ZK flavours) to authenticate to a Verifier.
Req-IB#03	Verifier	Web application demonstrating TLS 1.3 server authentication and SSI client authentication with PQC.
Req-IB#04	Identity Framework	Library implementing identity management as in the SSI model, in compliance with DID [3] and VC [4] W3C specifications.
Req-IB#05	JWT- & JPT-Based Authentication	VPs and proofs MUST be exchanged in JWT [5] and JPT [6] formats.
Req-IB#06	Cryptographic Algorithms	Traditional and PQ implementation of algorithms for digital signatures and ZKPs suitable for SSI management.
Req-IB#07	Quantum-Secure TLS Channel	PQ TLS 1.3 channel between Holder & Issuer, and Holder & Verifier. The TLS channel SHOULD ensure server authentication via PQ/T Hybrid PKIX certificates.
Req-IB#08	Platform Support	The main development and testing platform for the module and the library integrations is Linux.
Req-IB#09	Server Instances	Web server deployment infrastructure. A server instance in Europe and a server instance outside (presumably in USA). The instances SHOULD have both IPv4 and IPv6 connectivity. The instances SHOULD support multiple clients simultaneously.
Req-IB#10	Server Domains	Subdomains. Several 3rd-level (and below, if necessary) subdomains SHOULD be created under the top project domain `qubip.eu` (e.g., `test1.qubip.eu`, etc). Different subdomains SHOULD be dedicated for different algorithm setups.

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

Req.ID	Name	Description
Req-IB#11	Server Container	Implement and maintain a Fedora Linux container suitable for running the traditional and PQ/T-capable TLS server with minimal end-user setup.
Req-IB#12	Client Instances	Implement and maintain a repository for installation of all client applications (Mozilla Firefox, Digital Wallet) and necessary libraries for running the pilot.
Req-IB#13	Client Live Image	Implement a Linux live image suitable for running the client-side traditional and PQ/T-capable applications with minimal end-user setup.
Req-IB#14	TLS Connection Generator Tool	Software to automatically open TLS connections to the servers and maintain a given level of load to test the system with different traffic loads at server side.
Req-IB#15	Server Documentation	Documentation covering the steps to setup and run all instances of server applications.
Req-IB#16	Client Documentation	Documentation covering the steps to setup and run all instances of client applications and connect to servers.
Req-IB#17	Benchmark Datalake	Representation of the collected telemetry data and provide the collected data in a well-known format suitable for data analysis.
Req-IB#18	Interoperability	MbedTLS-based TLS clients (IoT nodes in chapter 2) MUST be compatible with OpenSSL-based TLS servers.
Req-IB#19	Informed Consent	Informed consent from the users involved in the tests in compliance with GDPR.
Req-IB#20	Education & Awareness	Provide users with information and resources to understand the importance of PQC and the changes it brings to their online experience. Offer guidance on new authentication methods, especially around the use and management of ZK VCs.
Req-IB#21	User Feedback	Design user interfaces that clearly communicate the security features enabled by PQC, without overwhelming the user with technical details. Ensure that the additional steps or computational time required by PQC are minimized or clearly justified to the user to maintain a positive UX.
Req-IB#22	Preconfigured Linux OS	Provide a preconfigured Fedora Linux-based client live image for the end-users to participate in the demonstrator. This is intended to simplify the process and resources required, allowing practically any personal computer or computers in computer rooms to be used for testing.

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

Req.ID	Name	Description
Req-IB#23	Playbook for End-Users	Develop a detailed playbook that includes: • Instructions for using the preconfigured Linux OS live image. • Steps for participating in the pilot, including how to perform tasks and use PQC-enabled browsers. • Guidance on providing feedback and reporting issues.

3.2 Specifications

Table 3.2 shows the specifications of the pilot demonstrator, based on Table 3.1 which shows the requirements. Each specification is associated with a particular requirement, which is listed in the **Req.ID** column. Most requirements have more than one associated specification, but each specification is only associated with one requirement.

Table 3.2: Internet Browsing Pilot specifications

Spec.ID	Name	Description	Req.ID
Spec-IB#01	User Registration	Implement a user registration form to insert the subject information used for issuing the VC. Implement backend logic for VC creation.	Req-IB#01
Spec-IB#02	User Identity Management	Users can authenticate themself using their VC. Create a user profile page where users can view, update, and revoke their VC. Implement backend logic to update or revoke a VC.	Req-IB#01
Spec-IB#03	Backend APIs	Backend exposes RESTful APIs to interact with the frontend. APIs is well-documented. Use Spec-IB#15 for SSI-related operations.	Req-IB#01
Spec-IB#04	Logging and Monitoring	Backend logs relevant events and errors for troubleshooting.	Req-IB#01
Spec-IB#05	Database	Backend uses a relational or NoSQL database to store application data.	Req-IB#01
Spec-IB#06	Identity Storage	The extension must store VCs, DIDs and private keys using IndexedDB. Encrypted at rest.	Req-IB#02
Spec-IB#07	User DID Operations	User can create, update, and deactivate DIDs. Use Spec-IB#15 for previous operations.	Req-IB#02
Spec-IB#08	User VC Operations	The user can select a desired VC and present it to a Verifier to access protected services. The Digital Wallet must support both cleartext and ZK VCs. Use Spec-IB#15 for SSI-related operations.	Req-IB#02

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

Spec.ID	Name	Description	Req.ID
Spec-IB#09	User Authentication	User can test both non-authenticated and authenticated services. User can use the Digital Wallet to present VCs, which the Verifier verifies before granting access to a service. Implement backend logic for VCs verification.	Req-IB#03
Spec-IB#10	Feedback Page	Page to indicate the security parameters of the TLS 1.3 PQ/T Hybrid connection and the disclosed identity information for feedback purposes.	Req-IB#03
Spec-IB#11	Survey Page	Web form to collect qualitative indicators of the tested functionalities from the users.	Req-IB#02
Spec-IB#12	Backend APIs	Backend exposes RESTful APIs to interact with the frontend. APIs is well-documented. Use Spec-IB#15 for SSI related operations.	Req-IB#03
Spec-IB#13	Logging and Monitoring	Backend logs relevant events and errors for troubleshooting.	Req-IB#03
Spec-IB#14	Database	Backend uses a relational or NoSQL database to store application data.	Req-IB#03
Spec-IB#15	Framework Selection	The IOTA Identity framework, specifically designed to comply with the DID and VC specifications. IOTA Identity is the target framework for PQC transition.	Req-IB#04
Spec-IB#16	JWT/JPT Encoding	JWT encoding is provided by IOTA Identity framework Spec-IB#15. JPT encoding is provided by json-proof-token library.	Req-IB#05
Spec-IB#17	Algorithms	Traditional signature algorithm: Ed25519 implementation provided by iota-crypto in IOTA Identity framework Spec-IB#15. PQ signature algorithms: NIST algorithm implementations provided by liboqs. Traditional ZK algorithm: BBS+. PQ ZK algorithm: BLNS. Both provided by ZKryptium library.	Req-IB#06
Spec-IB#18	PQ/T TLS 1.3	The OpenSSL and Mozilla Firefox TLS stacks supports negotiating and using PQ/T Hybrid key exchange and (server-side) authentication. The protocol flow is fully compliant with TLS 1.3 (RFC-8446 [7]). The protocol ensures server authentication via PQ/T Hybrid PKIX certificates.	Req-IB#07
Spec-IB#19	Flexible Algorithm Selection	The loadable modules framework for the integration of PQC in TLS 1.3 is flexible enough to allow Holder, Issuer, and Verifier to select their choice of PQ/T schemes and backing implementations.	Req-IB#07

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

Spec.ID	Name	Description	Req.ID
Spec-IB#20	Crypto Libraries	Implement the software (OpenSSL, OpenSSL `Provider`, low-level library providing PQ algorithms, NSS, Mozilla Firefox) supporting PQ algorithms for Linux platform.	Req-IB#08
Spec-IB#21	Software Repositories	Some components (liboqs and oqsprovider) are already included within standard Fedora Linux repos. Also OpenSSL, nginx, curl are a part of Fedora Linux and are available for installation using standard procedures. The other QUBIP-specific software (a specific PQ-capable version of NSS/Mozilla Firefox, QUBIP PQ `Provider`, VC backend, VC Mozilla Firefox plugin) may or may not be available on default Fedora Linux repos depending on the QUBIP consortium decision. These software components are available for installation through standard procedures from 3rd-party Fedora Linux-compatible repositories according to the FESCo Third-Party Repository Policy [8].	Req-IB#12
Spec-IB#22	Web Server	4 web servers in (Spain, Italy, Finland and Czechia) + 1 (outside EU) PQ/T web servers (Nginx-based) to test different network conditions and request loads.	Req-IB#09
Spec-IB#23	IP Stacks	Podman is the chosen containerization solution, which natively supports both IPv4 and IPv6 networking.	Req-IB#09
Spec-IB#24	Test Subdomains	Web servers configured to serve a different virtual SNI server for each tested PQ/T Hybrid configuration. They are subdomains of qubip.eu.	Req-IB#10
Spec-IB#25	Server Container	Implement and maintain a Linux container based on Fedora Linux, powered by Podman, which runs a traditional and PQ/T-capable TLS server. It requires minimal end-user setup.	Req-IB#11
Spec-IB#26	Client Live Image	Implement and maintain a Fedora Linux-based live image suitable for running the traditional and PQ/T-capable applications with minimal end-user setup.	Req-IB#13
Spec-IB#27	Server Documentation	Document detailing internal deployment instructions for the web server to ensure consistent deployments and avoid negative effects on the end-user experience.	Req-IB#15
Spec-IB#28	Client Documentation	Documentation aiming to minimize the manual configuration required for end-users, but providing instructions on how to tune some parameters, as needed.	Req-IB#16

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

Spec.ID	Name	Description	Req.ID
Spec-IB#29	TLS Connection Generator Tool	Use established existing software to generate artificial TLS loads. Evaluate which tools serve us better during the specification of the tests. Among the options, we preliminarily include Apache Bench and similar software.	Req-IB#14
Spec-IB#30	Data Benchmarking	A set of publicly available files in CSV format containing various measurements of connection/data exchange using various PQ, Hybrid, and classic algorithms.	Req-IB#17
Spec-IB#31	Interoperability	The result is dependent on the standard publication timeline. The implementations in one subpackage are interoperable. The implementations from different subpackages are interoperable. There is no obligation about interoperability with external platforms, but the interoperability testing is performed and the results are available.	Req-IB#18
Spec-IB#32	Informed Consent Document	Draft a comprehensive informed consent document that clearly explains the purpose of the pilot, the nature of the data being collected, how it will be used, and the rights of the participants, including their right to withdraw consent at any time.	Req-IB#19
Spec-IB#33	Compliance Check	Ensure the consent form complies with GDPR and other relevant data protection regulations.	Req-IB#19
Spec-IB#34	Educational Materials	Develop/curate educational materials including guides, FAQs, and videos that explain PQC, its benefits, and how it affects internet browsing.	Req-IB#20
Spec-IB#35	Education & Awareness Website	The QUBIP website hosts the educational resources.	Req-IB#20
Spec-IB#36	UI Design	Develop a landing web page for the experiments, exposing to the end-users the details of key exchange and authentication properties for the connection, with explicit indicators of PQ readiness level.	Req-IB#21
Spec-IB#37	Feedback Mechanism	Collecting end-user feedback on the demonstrator, using a web form survey interface in compliance with GDPR and other regulations.	Req-IB#21
Spec-IB#38	OS Selection	A live image based on Fedora Linux Atomic Desktops, customized with the software preconfigured to run the end-user-facing components of the demonstrator.	Req-IB#22

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

Spec.ID	Name	Description	Req.ID
Spec-IB#39	Playbook Content	The playbook includes: • Instructions on setting up the environment: booting the live image, software installation (if required), configuration and verification steps. • Instructions on the experiments: specific step-by-step tasks, and performance logging (if required). • Instructions on the feedback procedure: how to access surveys and questionnaires (e.g., link to online forms at logout time), and instructions for technical logs submission (if required). • Instructions for reverting configuration changes (if required). • Guidance on providing feedback and reporting issues. The playbook is delivered within the experiment web pages, in a convenient way that does not disrupt the execution of the experiments.	Req-IB#23

3.3 Architecture

The Internet Browsing pilot demonstrator adopts a classical client-server architecture.

In the experiments with the demonstrator, users will run a pre-configured Fedora Linux live image. Among the software installed on the live image, there is the web browser Mozilla Firefox and a browser extension that functions as a digital wallet, enabling users to use their self-sovereign identity for authentication purposes. Because all the necessary tooling for the client demonstrator is packaged in a pre-configured live image, users can run it on their own computers or on computers available at the experiment sites, without any specialized hardware.

PQ/T Hybrid cryptography–capable web servers will serve web content and allow clients to connect via Nginx, verifying the clients' digital identity. OpenSSL enables PQ/T Hybrid TLS by using the QUBIP provider to interface with a software implementation of the necessary algorithms for key exchange and authentication. The servers will be geographically distributed, to better simulate real-world conditions of Internet browsing. In addition to the testing done with human users interacting with the servers through Mozilla Firefox on the preconfigured live image, some automated testing and benchmarking will be performed to assess how the servers perform under heavy TLS traffic loads.

3.3.1 Client architecture

Figure 3.1 shows the client architecture for the pilot demonstrator. The client runs Fedora Linux.

At the top level, the Mozilla Firefox browser runs with an extension installed that functions as a digital wallet

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

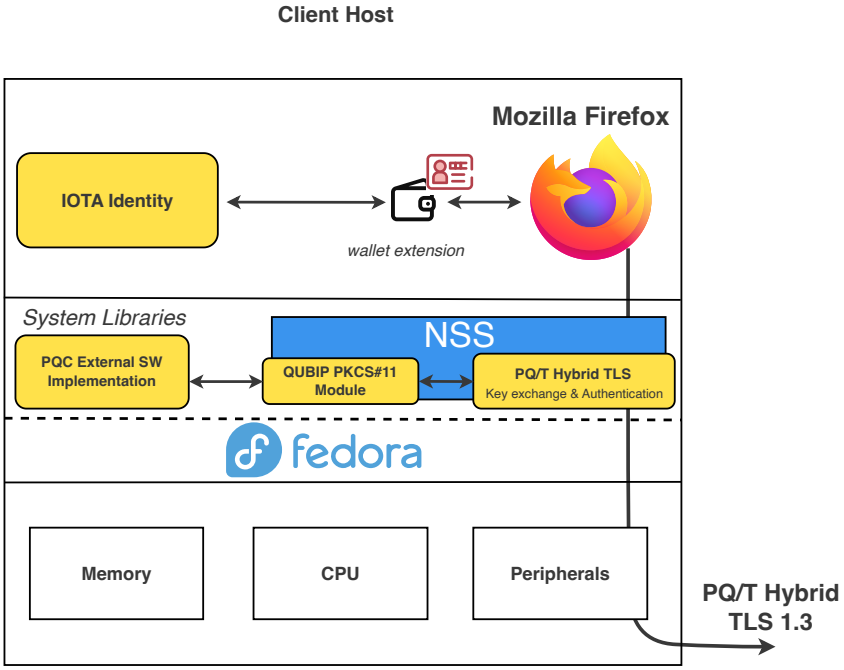


Figure 3.1: Client architecture of the Internet Browsing Pilot.

for storing and managing VCs. The wallet extension interacts with the IOTA Identity library, allowing the user to manage their self-sovereign identity.

At the middle level, Mozilla Firefox uses NSS to handle cryptographic operations. NSS provides PQ/T Hybrid key exchange and authentication in the TLS 1.3 protocol, and has a `PKCS #11` module which uses an external software library that implements PQC algorithms.

At the bottom level, the server communicates over the TLS 1.3 protocol using network peripherals.

3.3.2 Server architecture

Figure 3.2 shows the server architecture for the pilot demonstrator. The server runs Fedora Linux.

At the top level, a Nginx web server serves web content to clients, and authenticates incoming client requests by communicating with the Verifier. The Verifier uses the IOTA Identity library in the verification process of the client' Verifiable Presentation (VP) or Zero-Knowledge (ZK) proof.

At the middle level, Nginx is backed by the TLS functionality provided by OpenSSL. Through the extensible architecture of OpenSSL's provider model, external implementations (both in software and in hardware) of PQC algorithms are made available for use in the key exchange and authentication process. The QUBIP provider serves as the interface layer between OpenSSL and the external implementations. OpenSSL and Nginx are configured to support PQ/T Hybrid algorithms for key exchange and authentication.

At the bottom level, the server communicates over the TLS 1.3 protocol using network peripherals.

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

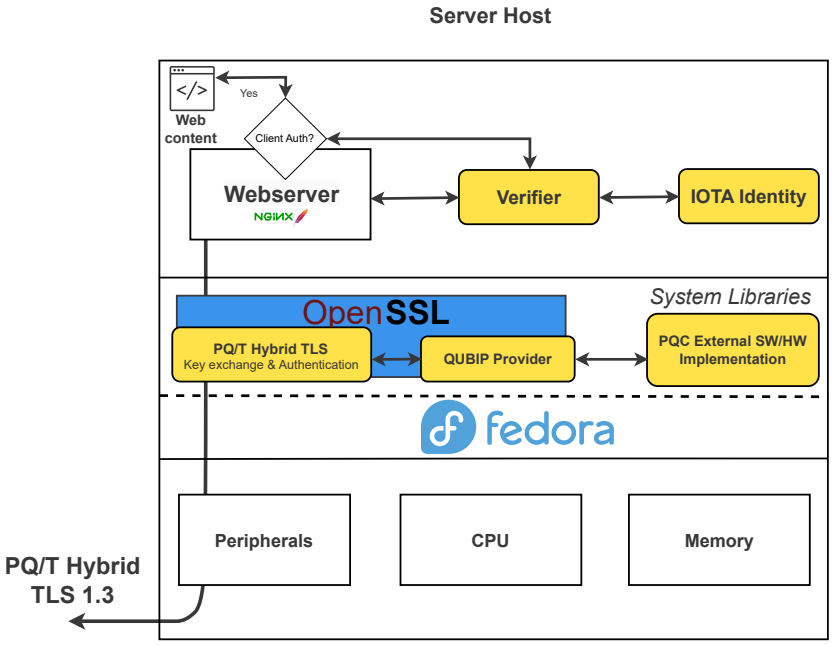


Figure 3.2: Server architecture of the Internet Browsing Pilot.

3.3.3 Client–Issuer interactions

Figure 3.3 shows the interaction between a Client that wishes to obtain a VC and an Issuer that issues VCs.

When requesting a VC from the Issuer, the Client already has a DID, and a key pair consisting of a secret key SK and a public key PK.

The Client and Issuer perform a TLS 1.3 handshake, establishing a secure communication channel, which provides server-side authentication using PQ/T Hybrid PKIX certificates. Then, the Client sends its DID and requests a challenge string CHALLENGE. The Issuer replies with the CHALLENGE. The Client computes a digital signature SIG over DID and CHALLENGE using his SK, and sends SIG to the Client, along with the Claim(s) they wish to have included in the VC. The Issuer resolves the DID, verifies the SIG using the Client's PK, and verifies the Claim(s). If all checks are successful, the Issuer creates the VC and sends it to the Client.

3.3.4 Client–Server interactions

Figure 3.4 shows the interaction between a Client and a Server that provides access to Hypertext Transfer Protocol over Secure Socket Layer (HTTPS) services to Clients who present a valid VP or ZK proof.

When contacting the Server, the Client has a full self-sovereign identity made of a key pair, the DID, and a VC.

The Client and Server perform a TLS 1.3 handshake, establishing a secure communication channel with server only authentication, using PQ/T Hybrid PKIX certificates. Then, the Client performs his authentication with the Server at application layer. The Client sends his DID and requests a challenge string CHALLENGE. The Server replies with the CHALLENGE. The Client creates a VP or ZK proof and sends it to the Server. The Server verifies it and responds to the Client to let it know whether the verification was successful or not. If the verification was successful, the Client now has access to web services.

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

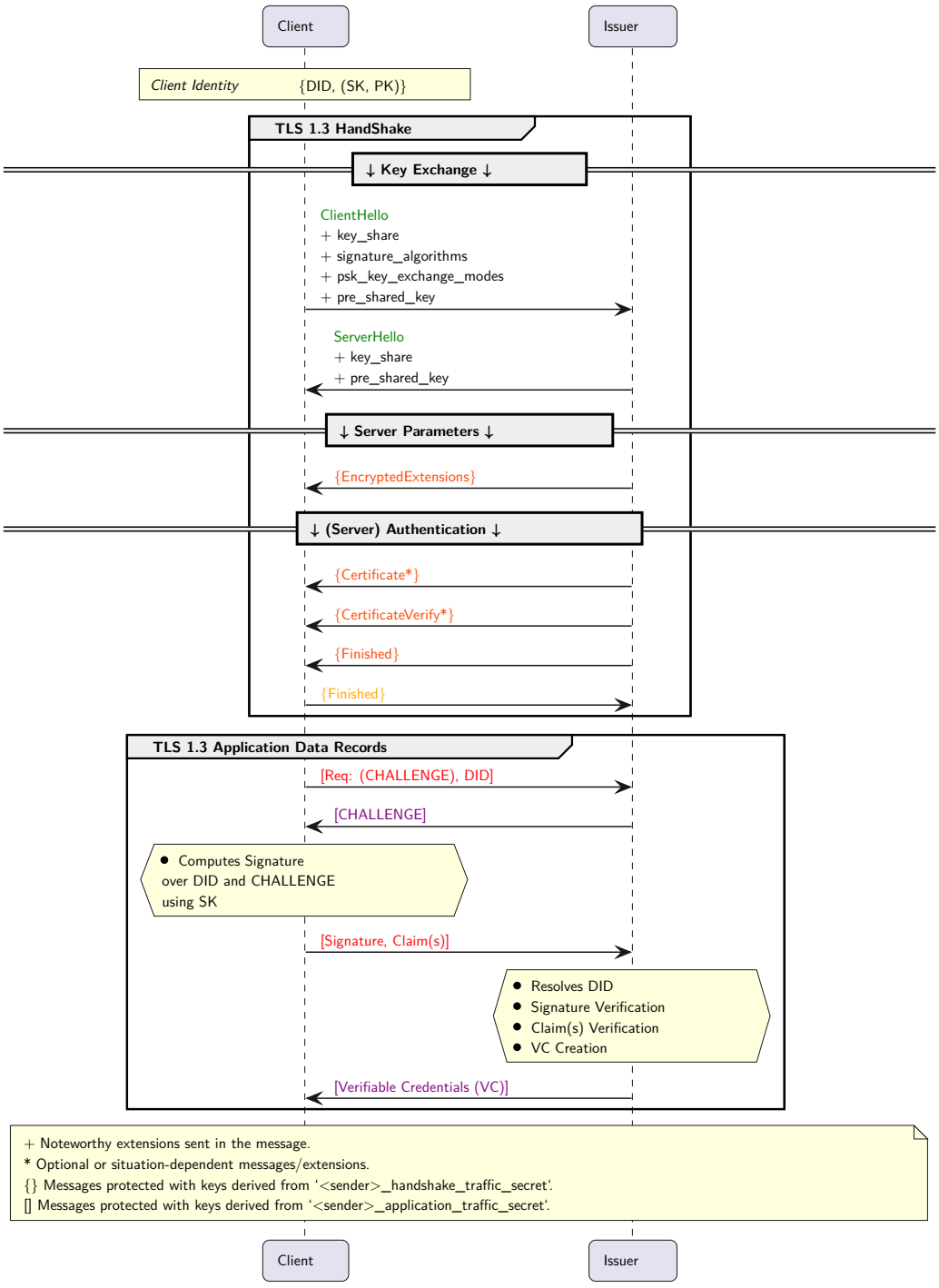


Figure 3.3: Sequence diagram of the interactions between Client and Issuer

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

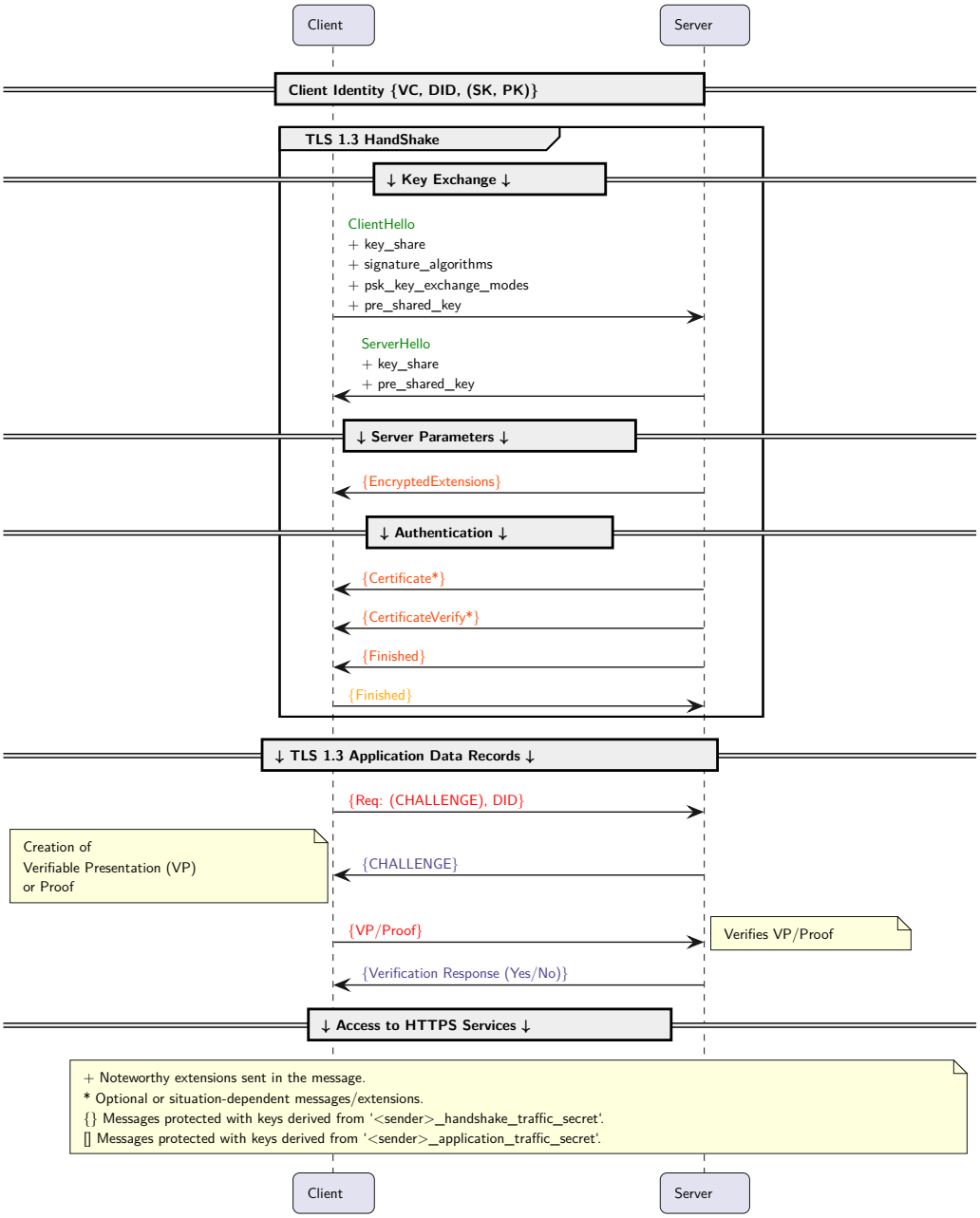


Figure 3.4: Sequence diagram of the interactions between Client and Server.

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

4 Quantum-secure Software Network Environments for Telco Operators

This pilot is designed to address the transition to PQC of the software environments adopted by telecommunications operators and network providers. Given the complexity of these environments, two key aspects have been the focus of attention. First, we must consider the fact of the massive adoption is happening around the new relevant technologies of network virtualization architectures, namely Network Functions Virtualization (NFV), and the ability to program the network, namely Software-Defined Networking (SDN). This type of network architecture is characterized by the decoupling of the control plane and the data plane, giving the ability to program the network control. Second, considering that these architectures are based in data centers, it is realistic to integrate complementary transition technologies like Quantum Key Distribution (QKD), which necessitates the implementation of physical systems that can be hosted within those data centers. Concerning the first aspect, the pilot demonstrator focuses on how to align this transition with reference solutions such as container-based virtualization and its secure interconnectivity in the data plane. Concerning the second aspect, the pilot demonstrator focuses on hybridization strategies between PQC and QKD, and on applying them programmatically to the interconnections of virtual nodes. Both aspects are addressed starting from the adoption of the Internet Key Exchange (IKE)-less IP Security (IPsec) building block designed in WP1 and addressed in Deliverable D1.4.

4.1 Requirements

The requirements are a set of demanded needs required to setup the software telco environment and are based on the assumption that the IKE-less IPsec building block from WP1 exists. These requirements are collected in Table 4.1.

The first five requirements (Req-TE#01 – Req-TE#05) identify the needs related to the architectures expected in telco environments. This includes consideration of the network virtualisation (NFV) and network scheduling (SDN) paradigms to make the transition to PQC compatible with existing solutions. Req-TE#06 and Req-TE#07 focus on the physical aspects of telco environments, i.e., physical perimeter protection (e.g., Data Centers), and the assumption that generic server models currently in use will continue to be necessary. Req-TE#08 emphasises the need to apply in-use standards wherever possible. Req-TE#09 aims to delimit the scope of the pilot demonstrator to the fundamental risk identified today, the data plane, especially when traffic circulates between security domains. Req-TE#10, Req-TE#11, and Req-TE#12 ensure the integration of QKD, identity management, and integrity verification in the pilot demonstrator as an integral part of the strategy for transition to PQC.

Table 4.1: Software Network Environments for Telco Operators Pilot requirements

Req.ID	Name	Description
Req-TE#01	Containers on Bare-metal	Native container deployment on bare metal needed for attestation compatibility, SHOULD avoid the use of virtual machines that cannot guarantee the attestation mechanism.
Req-TE#02	CNF-based Modules	All modules used in the pilot SHOULD be based on Container Network Function (CNF).

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

Req.ID	Name	Description
Req-TE#03	NFV Management	High level of management and orchestration for the deployment of the different CNFs SHOULD be managed using state-of-the-art solutions supported by Telcos.
Req-TE#04	SDN	SDN approach SHOULD be applied including a control plane post quantum transition aware for the data plane.
Req-TE#05	Overlay Network	Data plane connectivity between CNFs SHOULD happen in an overlay network that abstracts the quantum secure connectivity.
Req-TE#06	COTS	Solutions SHOULD be compatible with telco environment Commercial Off-the-Shelf (COTS) servers and standard architectures in cloud and data centers.
Req-TE#07	Environment Security	QKD system and COTS server SHOULD support logical and physical security equivalent to telco facilities environment.
Req-TE#08	Standardized Interfaces & Protocols	Standardized interfaces and protocols SHOULD be used when are available versus proprietaries ones, in management, control, and data plane.
Req-TE#09	Data Plane Security	Quantum risk of Store now and decrypt later attacks SHOULD be mitigated in the data plane Telco connectivity between CNFs.
Req-TE#10	QKD Connectivity	QKD links and associated data channel availability is needed to generate quantum secure keys.
Req-TE#11	Integrity Verification	Support of software based PQC solutions for integrity verification.
Req-TE#12	OCSP Verification	Digital certificates used in TLS protocol MUST be verified (by means of validity) by the TLS client through OCSP.

4.2 Specifications

Table 4.2 details the list of system specifications based on the requirements in Table 4.1. The Table explains how to meet each requirement through one or more specifications.

Table 4.2: Software Network Environments for Telco Operators Pilot specifications

Spec.ID	Name	Description	Req.ID
Spec-TE#01	Kubernetes	Kubernetes management solution using bare metal approach for attestation compatibility.	Req-TE#01
Spec-TE#02	CNFs based on Kubernetes Pods	Use of Pod compatible with Kubernetes management solution as CNFs.	Req-TE#02
Spec-TE#03	Open NFV MANO	Orchestration of the deployment of the different VNFs is managed using ETSI NFV Management and Orchestration (MANO) approach and Open Source MANO.	Req-TE#03

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

Spec.ID	Name	Description	Req.ID
Spec-TE#04	Open SDN	PQ transition management triggered by a SDN application approach, over a open source SDN controller (TFS, ONOS, etc.).	Req-TE#04
Spec-TE#05	Node Connectivity	Connectivity between different Kubernetes nodes uses a IPsec quantum secure solution at node level.	Req-TE#05
Spec-TE#06	CNF Connectivity	Connectivity between CNF or pods in different Kubernetes nodes use an overlay network provided by Link-Layer Secure connectivity for Microservice platforms (L2S-M) software switch on demand.	Req-TE#05
Spec-TE#07	COTS HW Telco Servers	Servers in the pilot are similar to the ones used in telco environment in term of resources (RAM, cores, network interfaces), type (rack mounted) and connectivity with physical switches.	Req-TE#06
Spec-TE#08	COTS Processors	Intel x86 architecture, alternatively others can be considered.	Req-TE#06
Spec-TE#09	Linux System	Telco environment COTS servers are deployed using standards Linux distributions as operative system and supporting Linux kernel IPsec.	Req-TE#06
Spec-TE#10	Security	Telco physical environment is in place, including security mechanism, such as physical access control and monitoring for QKD system and COTS server.	Req-TE#07
Spec-TE#11	Key Delivery	Key delivery between components is based on ETSI-004 standard.	Req-TE#08
Spec-TE#12	Control Plane Configuration	Control plane configuration for IPsec based on I2NSF interface (RFC-9061 [9]).	Req-TE#08
Spec-TE#13	NFV Deploy	Control plane deployment of CNF based on ETSI NFV SOL005 [10]	Req-TE#08
Spec-TE#14	Ciphering Algorithms	Configurable on demand type and length symmetric algorithms supported by IPsec Encapsulating Security Payload (ESP) based on AES.	Req-TE#09
Spec-TE#15	Integrity Algorithms	Configurable on demand type and length hash algorithms supported by IPsec ESP based on SHA2.	Req-TE#09
Spec-TE#16	Key Exchange Algorithms	Configurable on demand type and length KEM algorithms, including ML-KEM.	Req-TE#09
Spec-TE#17	Hybridization Agility	Configurable on demand type and length for hybridisation options (classical, PQC, QKD) for Key exchange to be used for CNF communication over IPsec.	Req-TE#09

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

Spec.ID	Name	Description	Req.ID
Spec-TE#18	QKD	QKD system (pair) or emulated one is available and operative between 2 Kubernetes (K8s) nodes or clusters.	Req-TE#10
Spec-TE#19	Free-Space Channel Link	Implementation of a free-space optical channel for QKD, with a horizontal alignment and a maximum distance of 300 meters.	Req-TE#10
Spec-TE#20	Quantum Data Laser Specifications	Utilization of an 850nm laser with 0.001mW power for transmitting quantum data between the QKD system's transmitter and receiver.	Req-TE#10
Spec-TE#21	Aiming Laser Specifications	Employment of a 785nm laser with 100mW power for aiming and synchronization purposes within the QKD system, facilitating accurate alignment of the free-space link.	Req-TE#10
Spec-TE#22	Error Correction and Key Distillation	Implementation of error correction algorithms to distil the quantum key from the raw key material obtained through the QKD process.	Req-TE#10
Spec-TE#23	Secure Boot	Verify signature over firmware and OS and eventually stop boot in case of failure.	Req-TE#11
Spec-TE#24	Measured Boot	Measure and securely store in Root of Trust for Storage (RTS) all software components from boot up to OS.	Req-TE#11
Spec-TE#25	TPM	HW TPM2.0 chip compliant with TCG standard to provide cryptographic identity, RTS and Root of Trust for Reporting (RTR) in classical algorithms.	Req-TE#11
Spec-TE#26	Remote Attestation	RA protocol for device enrolment and periodic attestation is used.	Req-TE#11
Spec-TE#27	Attestation Agent	Agent running on COTS server for device enrollment and responding to attestation requests.	Req-TE#11
Spec-TE#28	Attestation Service	Trust manager for COTS server enrollment, golden DB management, periodic attestation, status reporting.	Req-TE#11
Spec-TE#29	IMA	Linux version with support IMA module.	Req-TE#11
Spec-TE#30	Integrity PQ Algorithms	Combine TPM with software PQC algorithms over the assumption of physical security (Req-TE#07).	Req-TE#11
Spec-TE#31	OCSP	The PKI implements an OCSP responder to provide real-time certificate validation.	Req-TE#12

4.3 Architecture

The system is developed to secure the communications between two or more Pods/virtualised workloads in K8s clusters. The overall architecture, depicted in Figure 4.1, consists of a K8s cluster with the L2S-M solution already installed. This K8s operator (i.e., a software able to extend K8s management function-

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

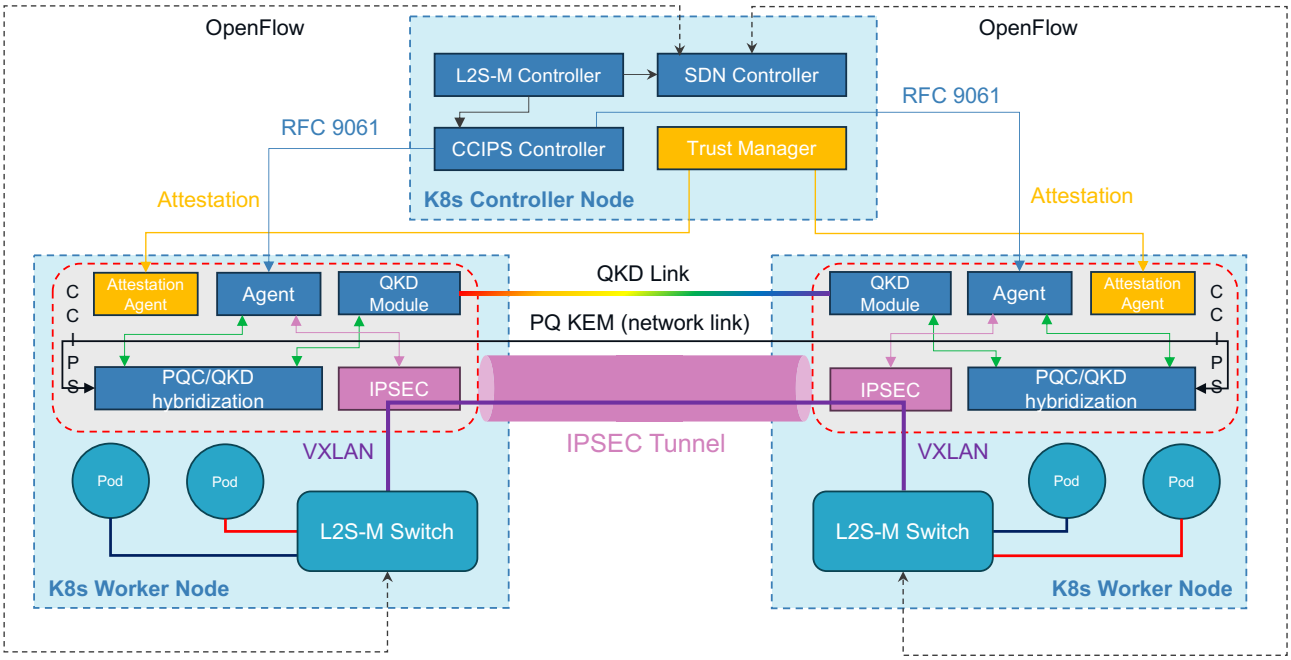


Figure 4.1: Architecture of the Software Network Environments for Telco Operators Pilot.

alities [11]) is responsible for enabling the creation of virtual networks that isolate traffic between Pods on K8s clusters. In addition, the L2S-M interacts with the Centrally Controlled IPSec (CCIPS) module to configure IPsec tunnels between nodes of the K8s cluster, and to ensure that the communications between Pods/virtualised workloads use a quantum-secure channel. Overall, the system provides isolated and secure communications all over the K8s cluster. The architecture can be split in two different sections related to the functionalities in the K8s Controller Node and in K8s Worker Nodes.

The elements in the K8s Controller Node (i.e., where all K8s management and control functionalities are located) are the following:

- L2S-M Controller: is in charge of managing virtual networks within the cluster, including their creation, management and deletion. Furthermore, this element signals the deployment of the switching functionalities within the K8s Worker Nodes to take advantage of the IPsec tunnels for enabling secure communications.
- SDN Controller: configures the forwarding tables of the virtual switches in the K8s Worker Nodes. This configuration, based on the OpenFlow protocol specification [12], is triggered from the L2S-M Controller based on the Pods included in the virtual network.
- CCIPS Controller: is in charge of coordinating the CCIPS operations in the system. It interacts with the L2S-M Controller to start the setup of IPsec tunnels between the K8s Worker Nodes.
- Trust Manager: coordinates the authentication and integrity verification of the software components in the K8s Worker Nodes.

The elements in the K8s Worker Nodes (i.e., the nodes that are used to deploy the K8s Pods/virtualised workloads in a K8s cluster) are the following:

- L2S-M Switch: is a virtual switch in charge of forwarding traffic to/from the Pods based on the configurations sent from the L2S-M controller The virtual switch includes a VXLAN [13] interface to connect to other L2S-M switches located in other K8s Worker Nodes.
- CCIPS Module: it includes the functionalities to establish quantum-secure IPsec tunnels between the K8s Worker Nodes. The full design of this module is described in Deliverable D1.4.

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

4.3.1 Sequence of interactions

Figures 4.2 and 4.3 shows the interactions between the elements of the pilot architecture. The sequence diagrams show two main processes: L2S-M installation, and L2S-M CNF deployment.

L2S-M installation

The first process consists on the installation of the L2S-M operator in the K8s cluster. These are the four steps of the process:

1. L2S-M operator installation in the K8s Controller Node: the platform owner starts the installation process by deploying the L2S-M Kubernetes operator and the SDN Controller in the K8s node. This installation is performed with the standard command line interface of K8s.
2. Deploy & configure L2S-M overlay: L2S-M proceeds to configure the overlay used to communicate between the Pods deployed in the different K8s Worker Nodes.

   To do so, L2S-M reads the overlay configuration and sends an instruction to the K8s Worker Nodes to deploy their respective L2S-M virtual switches. Furthermore, the L2S-M Controller instructs the K8s Worker Nodes to create and configure the VXLAN endpoints to establish the communication between the nodes, and attach the endpoint to the L2S-M virtual switch. As a result, the L2S-M operator is able to isolate the traffic exchanged between Pods, regardless of whether they are on the same or different nodes within the K8s cluster. At this stage, L2S-M provides a solution for isolating communications by using the overlay, but it does not yet provide secure communications.
3. Configuration of quantum-secure IPsec tunnel between K8s Worker Nodes: once the L2S-M deployment and configuration is completed, the L2S-M Controller signals the CCIPS Controller to start the configuration of the IPsec tunnel between the specified K8s Worker Nodes, using keys that are a proper hybridization of keys generated by PQ KEM and QKD as detailed in Deliverable D1.4. The CCIPS Controller takes care of interacting with Trust Manager to verify identities and confirm the valid state of the K8s Worker Nodes.
4. Setup of quantum-secure IPsec tunnel between K8s Worker Nodes: in this step, the CCIPS Controller establishes the IPsec tunnel. The steps involved are detailed in Figure 4.3. When the process is complete, the CCIPS Controller notifies the L2S-M Controller that the quantum-secure IPsec tunnel has been successfully established between the selected nodes, completing the L2S-M installation process.

The procedure to create the quantum-secure IPsec tunnel to interconnect Pods/virtualised workloads is depicted in Figure 4.3. These are the involved steps:

1. Attestation request: upon receiving a quantum-link configure setup message from the L2S-M Controller, the CCIPS Controller identifies the nodes involved in each link (the L2S-M Controller may ask for the configuration of multiple links) and requests the Trust Manager to verify their trustworthiness.
2. Verify endpoint request: upon receiving the request, the Trust Manager interacts with the Attestation Agent in each node to verify their trustworthiness by means of a remote attestation protocol.
3. Verification confirmation: if both agents were successfully verified, and therefore authenticated, the Trust Manager sends a confirmation message to the CCIPS Controller to proceed with the IPsec tunnel setup between the trusted nodes.
4. IPsec tunnel request: the CCIPS Controller sends an IPsec tunnel requests towards each CCIPS Agent in the tunnel endpoint nodes.
5. Key request: both CCIPS Agents request to their local PQC/QKD Hybridisation module the corresponding keys. The keys can be generated and agreed in two different ways (or combining both methods):

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

- via the QKD Module in each node,
- via the PQC/QKD Hybridisation module using also PQ KEM algorithms.

Once the keys are generated, the PQC/QKD Hybridization module sends the key to their local CCIPS Agent.

6. IPsec Tunnel Activation: The CCIPS Agent proceeds to include the keys in the respective databases and activate the IPsec Tunnel.
7. Activated IPsec tunnel confirmation: finally, the CCIPS Agent in each endpoint node of the tunnel informs the CCIPS Controller that the tunnel has been correctly activated, enabling secure communications between nodes.

L2S-M CNF deployment

The second process consists on the deployment of CNF within the K8s cluster. With the assist of both CCIPS and L2S-M modules, the Pods deployed in different K8s Worker Nodes communicate in an isolated (provided by the link-layer virtual networks in L2S-M) and secure (provided by the quantum-secure IPsec tunnel between nodes) manner. Figure 4.2 shows the steps involved:

1. Create virtual network: the K8s administrator creates the virtual networks needed in the cluster, using K8s command line and registering them as a K8s resources in the cluster.
2. Deploy Pods in the virtual network: the user deploys the Pods that need to communicate through the virtual network. The L2S-M Controller assigns an interface in the L2S-M Switch in the K8s Worker Nodes, which in turn enables the deployment of the Pods attached to that interface (the deployment was triggered by the K8s Controller).
3. Configure virtual switch: once the Pods were deployed and attached to the corresponding interfaces in the L2S-M Switch, the L2S-M Controller signals the SDN Controller to configure the forwarding tables of switches on the cluster, using the OpenFlow 1.3 protocol [12] to connect all Pods in the virtual network, thus enabling their isolated communications.
4. Secure communication between Pods: Since all communications links in the overlay are secured by the CCIPS module, Pods can communicate among themselves in a secure manner, while isolating their data planes from the rest of Pods in the K8s cluster.

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

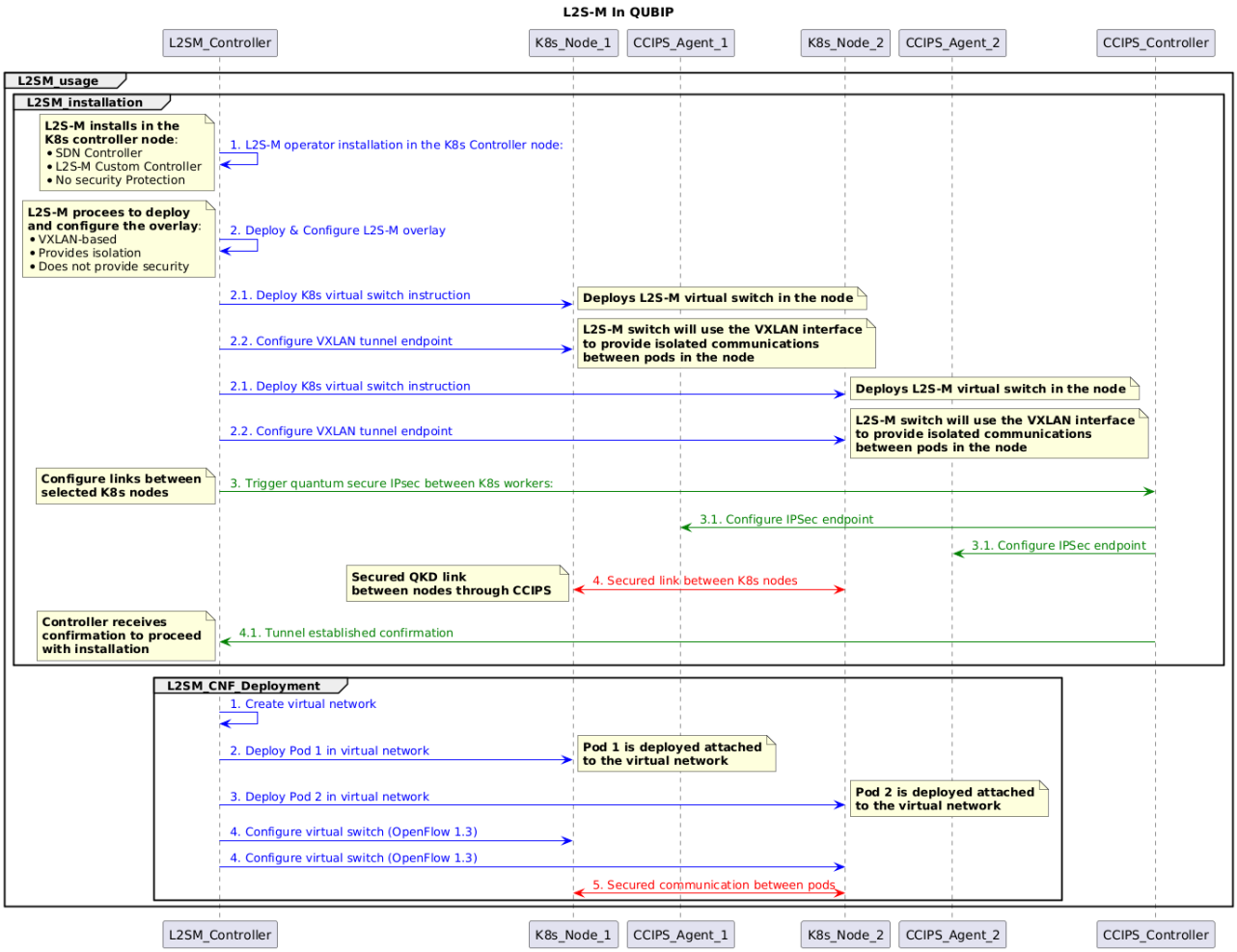


Figure 4.2: Sequence diagram of the procedures for L2S-M installation and L2S-M CNF deployment.

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

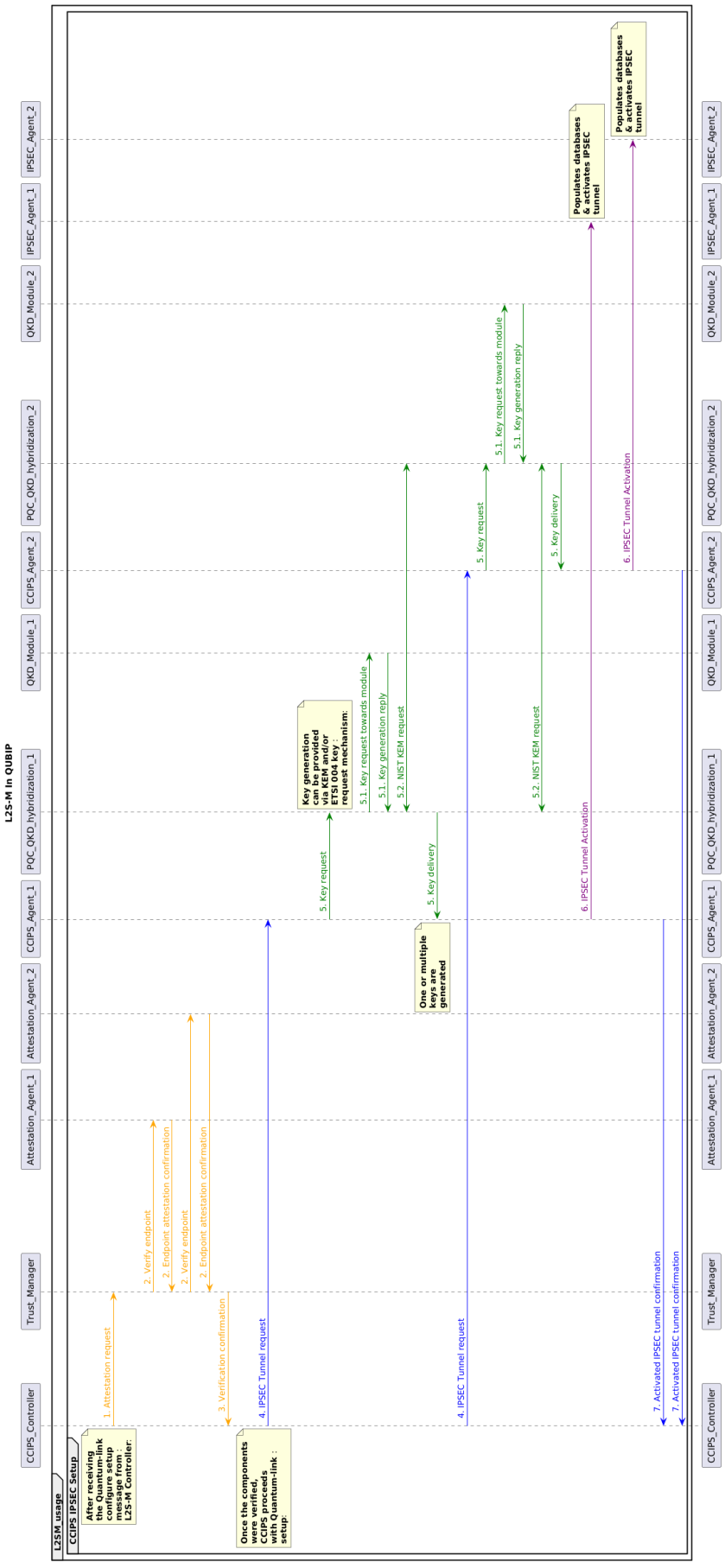


Figure 4.3: Sequence diagram of the procedure for quantum-secure IPsec tunnel setup.

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

5 Conclusions

This document describes the three pilot demonstrators of the QUBIP project, detailing the system requirements and specifications, and the overall architectures. These pilot demonstrators will be built by integrating the eight PQ building blocks designed and developed in WP1. The integration of the building blocks will generate a cascade of issues that are an integral part of the transition process to PQC. All issues will be promptly resolved and documented to produce a knowledge base, one of the main outputs of the QUBIP project.

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

Bibliography

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

[2] Trusted Computing Group, “TCG EFI Platform Specification Version 1.22”, January 2014, https://trustedcomputinggroup.org/resource/tcg-efi-platform-specification/

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

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

[5] M. B. Jones, J. Bradley, and N. Sakimura, “JSON Web Token (JWT)”, RFC-7519, May 2015, DOI 10.17487/RFC7519

[6] M. B. Jones, D. Waite, and J. Miller, “JSON Proof Token”, Internet Draft RFC, October 2024, https://datatracker.ietf.org/doc/draft-ietf-jose-json-proof-token/07/

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

[8] Fedora Documentation, “FESCo Third-Party Repository Policy”, 2024, https://docs.fedoraproject.org/en-US/fesco/Third_Party_Repository_Policy

[9] 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

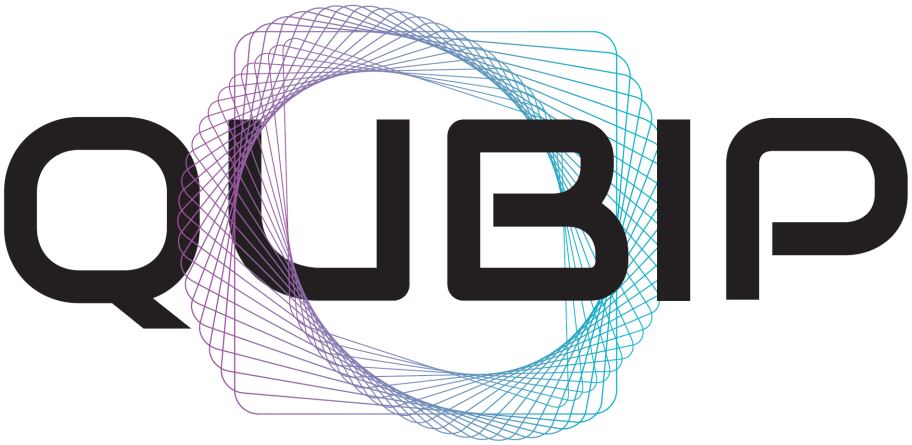
[10] ETSI, “Network Functions Virtualisation (NFV) Release 4; Protocols and Data Models”, 2022, https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/04.03.01_60/gs_nfv-sol005v040301p.pdf

[11] Kubernetes Documentation, “Operator Pattern”, July 2024, https://kubernetes.io/docs/concepts/extend-kubernetes/operator/

[12] Open Networking Foundation, “OpenFlow Switch Specification Version 1.3.0”, June 2012, https://opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[13] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, “Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks”, RFC-7348, August 2014, DOI 10.17487/RFC7348

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

D2.1 – Specifications of Pilot Demonstrators

Version 1.0

Horizon Europe