



PiQASO



PiQASO D2.1 Operational Landscape, Requirements, PQ Conformity Assessment and PQC Service Modalities - Initial Version

Grant Agreement	101190366
Project Title	Post-Quantum Cryptography As-a-Service for Common Transmission Systems and Infrastructures
Project acronym	PiQASO
Project Start Date	01 January 2025
Number of Deliverable	D2.1
Report Title	Operational Landscape, Requirements, PQ Conformity Assessment and PQC Service Modalities - Initial Version
Related Work Package	WP2
Related Task	T2.1, T2.2, T2.3, T2.4
Lead Organisation	UBITECH
Submission Date	30.06.2026
Last Change Date	20.07.2026
Dissemination Level	Public

Abstract

D2.1 presents the PiQASO architecture and its application to the post-quantum (PQ) transition of nine application domains: Automotive, Automation, Finance, Energy, Healthcare, Aerospace, Online Media, UAVs, and Transportation. Building upon the technical foundations established in D3.1, the architecture positioning PiQASO components across the PQ transition workflow is defined. For each application domain, D2.1 details the PQ transition pathway, defines Reference Scenarios and User Stories to guide experimentation, and identifies the relevant PiQASO PQC Ensemble functionalities required. Overall, D2.1 establishes the foundation for the evaluation of PiQASO across all planned demonstrators and defines the workflows that will govern subsequent experimentation activities within the project testbed.

Authoring & approval

Author(s) of the document

Name (Organisation name)	Date
Dimitris Karras (UBI)	09.07.2026
Thanasis Giannetsos (UBI)	09.07.2026
Stylianos Kazazis (QUBI)	08.07.2026
Niovi Sassalou (QUBI)	08.07.2026
Dimitris Katsinis (QUBI)	08.07.2026
Stelios Vourakis (ACCELI)	30.06.2026
Spyros Boletis (ACCELI)	30.06.2026
Dimitris Perikleous (ACCELI)	30.06.2026
Stefanos Chamalelis (DPG)	30.06.2026
Zampoglou Zoi (DPG)	30.06.2026
Konstantina Ferentinou (DPG)	30.06.2026
Sofia Kontopiraki (DPG)	30.06.2026
Stavros Maragkoudakis (DPG)	30.06.2026
Gavriil Despina (DPG)	30.06.2026
Utkarsh Goyal (TLU)	30.06.2026
Arnor Solberg (TLU)	30.06.2026
Maria Papadopoulou (MOH)	30.06.2026
Michela Cavuoto (PAL)	30.06.2026
Martí Fabregat (CXB)	30.06.2026
Ramon Martín de Pozuelo (CXB)	30.06.2026
Dimitrios Karanassos (CERTH)	30.06.2026
Aristeidis Apostolidis (CERTH)	30.06.2026
Dimosthenis Ioannidis (CERTH)	30.06.2026
Konstantinos Votis (CERTH)	30.06.2026
Anastasios Drosou (CERTH)	30.06.2026
Vlad Constantinescu (NEDHO)	30.06.2026
Adrian Obae (NEDHO)	30.06.2026
Marco Manso (PARTICLE)	30.06.2026
José Pires (PARTICLE)	30.06.2026
Bárbara Guerra (PARTICLE)	30.06.2026
Nikos Fotiou (STROWL)	30.06.2026
Savvas Argyropoulos (STROWL)	30.06.2026
Panagiotis Filntisis (STROWL)	30.06.2026
Konstantin Klein (BIBA)	30.06.2026
Gabriele Carru (ABI)	30.06.2026
Giuseppe Meloni (ABI)	30.06.2026
Fabio Piccolo (ABI)	30.06.2026

Maria Katiusia Zedda (ABI)	30.06.2026
Carolina Tomás Alfonso (FGC)	30.06.2026

Reviewed by

Name (Organisation name)	Date
(BYTE)	06.07.2026
Spyros Mazarakis (UNIS)	06.07.2026
Konstantina Liagkou (UNIS GR)	06.07.2026

Approved for submission

Name (Organisation name)	Date
Savvoula Oikonomou, UBITECH	20.07.2026

Document history

Version	Date	Contributor/Organisation	Additional information
0.1	30.06.25	UBI, QUBI	V01 created, ToC prepared
0.2	30.09.25	QUBI, UBI, All Use Case partners	V02 initial input on UC descriptions, sequence diagrams prepared. Weekly calls with technical partners for feedback and development.
0.3	29.01.26	All Use Case partners	V03 with further input on Chapters 6 – 14. Weekly calls with technical partners for feedback and development.
0.4	30.03.26	All Use Case partners	V04 with further input on Chapters 6 – 14, specification of requirements and per scenario sequence diagrams. Weekly calls with technical partners for feedback and development.
0.5	30.04.26	UBI, QUBI	V05 with input on Chapters 1 – 5. Weekly calls with technical partners for feedback and development.
0.6	05.06.26	UBI, QUBI, All Use Case partners	V06 finalisation of content for plenary meeting review. Weekly calls with technical partners for feedback and development.
0.7	30.06.26	UBI, QUBI, All Use Case partners	V07 review ready version with feedback from plenary addressed.
1.0	20.07.26	UBI, QUBI	V1 ready for submission with review comments addressed.

Copyright statement © 2025 – PiQASO Consortium
All rights reserved. Licensed to PiQASO under conditions.

PiQASO

Post-Quantum Cryptography As-a-Service for Common Transmission Systems and Infrastructures

The project funded under Grant Agreement No. 101190366 is supported by the European Cybersecurity Competence Centre



Co-funded by
the European Union



ECCE
EUROPEAN CYBERSECURITY
COMPETENCE CENTRE

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Cybersecurity Competence Centre. Neither the European Union nor the granting authority can be held responsible for them."



Executive Summary

Considering the advent of quantum computers, it is critical to ensure that existing information security systems currently applied in various application domains are able to cope with attackers and malicious entities who possess quantum computing capabilities. However, one core challenge in this regard is that existing service graph chains (SGCs) may consist of heterogeneous devices with varying computational capabilities, including legacy devices with limited ability to support the advanced mathematical capabilities required for executing complex post-quantum cryptographic operations. In this regard, one core target of PiQASO is to *provide the tools required for the transition of such SGCs to the post-quantum era, through the provision of the required tools and security solutions for ensuring that all entities remain at a trustworthy level throughout their operational lifecycle*. In this regard, D2.1 builds upon the initial definition of D3.1 and presents a comprehensive overview of the PiQASO architecture, and details its application to the post-quantum transition of a wide range of service graph chains across diverse application domains. Building upon the technical foundations and PQC migration rationale established in D3.1, we define a structured architecture that positions and integrates PiQASO components across heterogeneous target infrastructures, ranging from constrained tactical nodes to large-scale backend systems.

Chapter 2 defines the PiQASO architecture, detailing the component positioning and interactions that enable the post-quantum transition of target infrastructures in a structured and systematic manner. In Chapter 4, particular focus is placed on the **Crypto Asset Inventorying (CAI)** component, whose purpose is to build a structured, evidence-based understanding of the cryptographic and software-dependency posture of an operational environment before any post-quantum protection mechanisms is introduced. In this regard, we also provide detailed results of the application of the CAI in the context of the PiQASO use case demonstrators. Chapter 3 presents the requirements elicitation methodology adopted by PiQASO, which follows a top-down, defence-driven approach designed to ensure that the resulting framework requirements are grounded in operational reality and remain reusable across heterogeneous missions and infrastructures. This chapter also introduces the PiQASO Minimum Viable Product (MVP), which serves to demonstrate the common vision of the consortium and has been carefully aligned with the demonstrator activities scheduled throughout the project lifecycle, providing a coherent and well-structured foundation for all subsequent development and evaluation stages.

The core contribution of this deliverable lies in the detailed characterisation of the post-quantum transition pathways for nine application domains, each addressed in a dedicated chapter. Specifically, Chapter 5 addresses the **Automotive** domain, Chapter 6 the **Automation** domain, Chapter 7 the **Finance** domain, Chapter 8 the **Energy** domain, Chapter 9 the **Healthcare** domain, Chapter 11 the **Aerospace** domain, Chapter 13 the **Online Media** domain, Chapter 14 the **UAV** domain, and Chapter 15 the **Transportation** domain. These domains collectively represent a broad and heterogeneous set of service graph chains, spanning safety-critical embedded systems, mission-critical command and control networks, high-throughput financial infrastructures, and resource-constrained tactical deployments. For each domain, the deliverable provides a thorough description of the use case context and operational objectives, identifies the specific post-quantum transition requirements arising from the domain characteristics, and defines a set of Reference Scenarios and User Stories that will guide the experimentation activities conducted through the PiQASO testbed.

Finally, Chapter 16 consolidates the outcomes of the use case analysis into a structured overview of all PiQASO functionalities selected for integration across the application domains. For each domain, this chapter details the relevant components of the **PiQASO PQC Ensemble**, which encompasses the full suite of post-quantum cryptographic primitives and services offered by the framework, including Forward Security through Updatable Public Key Encryption (UPKE), Functional Encryption, as well as hardware-based accelerations and algorithmic optimisations. The chapter further describes the intended usage of each selected functionality within its respective use case context, and provides estimates of the expected implementation effort per domain.

Overall, D2.1 establishes the evaluation foundation for all planned PiQASO demonstrators and defines the concrete experimentation workflows that will govern subsequent activities within the project testbed, ensuring a coherent path from architectural design to practical validation.

Contents

1	Introduction	21
1.1	Relation to other WPs and Deliverables	22
1.2	Deliverable Structure	23
2	PiQASO Conceptual Architecture and Functional Components	25
2.1	Summary of PiQASO Conceptual Architecture	25
2.2	PiQASO PQC Software Development Kit (SDK)	28
2.2.1	PQC SDK Interface	28
2.2.2	Functional tests	29
2.3	PiQASO QR-TCB Architecture	29
2.3.1	Key Encapsulation Mechanisms	30
2.3.2	Digital Signature Algorithms	31
2.4	PiQASO Hardware Acceleration for PQC	32
2.4.1	Acceleration of Hash primitives	33
2.4.2	Acceleration of Polynomial Multiplication	33
2.4.3	Integration of the Accelerators into the ARM Platform	34
2.5	“PQC-as-a-Service” for Cloud-based Data Security	34
2.5.1	Initial Approach to Cloud Storage for Encrypted Data	35
3	Methodology for Elicitation of MVP, Requirements, and KPIs	44
3.1	Methodology for MVP Design	44
3.1.1	Requirements Definition Process	45
3.2	Requirements Elicitation Methodology	46
3.2.1	Technical Requirements Specification Process	46
3.2.2	Use Case Requirements Specification Process	47
3.3	Evaluation Plan and Application-Agnostic KPIs	48
3.4	Common PiQASO Operation Sequence Diagrams	55
3.4.1	PQ-TLS - Variant 1: Direct Client-Server Authentication	56
3.4.2	PQ-TLS - Variant 2: Authentication Through PQCaaS	57
3.4.3	Updatable Public Key Encryption	59
3.4.4	Attribute-Based Encryption	61

4	Cryptographic Asset Inventory Instantiation and Use-Case Specific SBOM	64
4.1	Hypothesis Establishment	68
4.2	Results per Use Case	69
5	PiQASO UC1 – Sector: Automotive	76
5.1	Use case's Description, Goals and Objectives	76
5.2	PiQASO Functionalities	78
5.3	Managed PQC-enabled vPKI Infrastructure	79
5.3.1	vPKI Architecture and Actors	80
5.3.2	Certificate Issuance Workflow	80
5.3.3	Certificate Lifecycle Management	82
5.4	"As-Is" Scenario	82
5.4.1	System Model and Current Security Architecture	82
5.4.2	Communication Workflow	85
5.4.3	As-Is vPKI Certificate Provisioning Workflow	87
5.4.4	Limitations of the Current Security Architecture	88
5.5	"To-Be" Scenario: PiQASO Architecture for Post-Quantum Security	89
5.5.1	System Model and Target Architecture	90
5.5.2	"To-Be" Reference Scenario #1: Quantum-Secure Emergency Vehicle Priority Management .	92
5.5.3	"To-Be" Reference Scenario #2: Quantum-Secure OTA Software Update	96
5.6	Reference Operational Scenarios	96
5.6.1	User Story #1 (Reference Scenario #1): Emergency Vehicle Priority Management	97
5.6.2	User Story #2 (Reference Scenario #2): Quantum-Secure OTA Software Update	100
5.6.3	User Story #3 (Reference Scenario #3): Complementary Scenario: Intersection Safety Coordination	105
5.7	Validation Strategy, Evaluation Metrics and Acceptance Criteria	105
6	PiQASO UC2 – Sector: Automation	107
6.1	Motivation and Challenges in the Automation Sector	107
6.1.1	Current Landscape: Autonomous Mobile Robots in Industrial Deployments	107
6.1.2	The Convergence of Cybersecurity and Physical Safety	108
6.1.3	The Transition from Classical to Post-Quantum Cryptography in Robotics	108
6.1.4	Motivation for PiQASO's Role in the Automation Sector	109
6.2	Use case's Description, Goals and Objectives	110
6.3	PiQASO Functionalities	110
6.3.1	Rationale for the Phased Implementation	111
6.4	"As-is" Scenario	111
6.5	Components and Roles Considered	112
6.6	Challenges and Needs from PiQASO	115
6.7	"To-Be" Reference Scenario #1: Secured ROS 2 Communications	115

6.8	“To-Be” Reference Scenario #2: Secured Fleet Manager with Selective Sharing and Protected Storage	116
6.9	“To-Be” Reference Scenario #3: ASSIST – Secure Remote Assistance	117
6.10	Reference Scenario User Stories and Metrics of Success	118
6.10.1	User Story #1 (Scenario #1): Secured Task Assignment	118
6.10.2	User Story #2 (Scenario #2): Protected Data Logging with Evolutionary Security	119
6.10.3	User Story #3 (Scenario #3): Secure Remote Intervention	120
6.10.4	Metrics of Success (KPIs)	121
7	PiQASO UC3 – Sector: Finance:	124
7.1	Use case’s Description, Goals and Objectives	124
7.2	PiQASO Functionalities	125
7.3	“As-is” Scenario	125
7.4	Components and roles considered in the context of Secure Online Payments Use Case	125
7.5	Challenges and Needs from PiQASO	127
7.6	“To-Be” Reference Scenario: PQC Secured Online Banking Operations	128
7.6.1	User Story #1 (Scenario #1): PQC Protection for External Communications (Edge Integration)	128
7.6.2	User Story #2 (Scenario #2): PQCaaS for External Communications (SaaS Model)	130
7.6.3	User Story #3 (Scenario #3): FPGA-Accelerated PQC Operations	131
7.6.4	Overall Architecture Perspective	133
7.6.5	Metrics of Success (KPIs)	134
8	PiQASO UC4 – Sector: Energy	135
8.1	Use Case’s Description, Goals and Objectives	135
8.2	PiQASO Functionalities	136
8.3	“As-Is” Scenario	137
8.4	Components and Roles Considered	137
8.5	Challenges and Needs from PiQASO	138
8.6	“To-Be” Reference Scenario #1: Quantum-Secure Smart Grid Telemetry	139
8.7	“To-Be” Reference Scenario #2: Crypto-Agile Gateway	140
8.8	“To-Be” Reference Scenario #3: Quantum-Secure Edge Device Onboarding	141
8.9	Reference Scenario User Stories and Metrics of Success	142
8.9.1	User Story #1 (Scenario #1): Secure Grid Telemetry Transmission	142
8.9.2	User Story #2 (Reference Scenario #2): Dynamic PQC Algorithm Switching (Crypto-Agility)	144
8.9.3	User Story #3 (Reference Scenario #3): Quantum-Secure Edge Device Onboarding	145
8.9.4	Key Performance Indicators (KPIs)	147
9	PiQASO UC5.1 – Sector: Healthcare - Remote Patient Care	149
9.1	Use Case’s Description, Goals, and Objectives	149
9.2	PiQASO Functionalities	153
9.3	“As-is” Scenario	154

9.3.1	Efficient and trustworthy onboarding	154
9.3.2	Trustworthy transmission of data	155
9.3.3	Robust and trustworthy scaling	157
9.4	Components and Roles Considered	158
9.5	Challenges and Needs from PiQASO	159
9.6	"To-Be" Scenarios	160
9.6.1	"To-Be" Reference Scenario #1: Onboarding and Device Authentication	161
9.6.2	To-Be" Reference Scenario #2: Trustworthy Transmission of Medical Data	161
9.6.3	To-Be" Reference Scenario #3: Robust and Trustworthy Software Updates	161
9.7	Reference Scenario User Stories and Metrics of Success	162
9.7.1	User Story #1 (Reference Scenario #1): Secure and Scalable Gateway Onboarding	162
9.7.2	User Story #2 (Reference Scenario #2): Confidential and Trustworthy Medical Data Trans- mission	164
9.7.3	User Story #3 (Reference Scenario #3): Trusted Software Update Deployment	166
9.7.4	Metrics of Success	168
10	PiQASO UC5.2 - Sector: Healthcare - Enhancing a Smart Ambulance System with PQC capabilities	170
10.1	UC Goals and Objectives	171
10.2	PiQASO Functionalities	172
10.3	Components and Roles Considered	172
10.4	"As-Is" Scenario	173
10.5	"To-Be" Reference Scenario #1: Secure authentication using PQ-TLS via PQCaaS	177
10.6	"To-Be" Reference Scenario #2: Secure sending and accessing patient data using PQ-TLS via PQCaaS	178
10.7	Reference Scenario User Stories and Metrics of Success	178
10.7.1	User Story #1 (Scenario #1): Secure User Authentication	178
10.7.2	User Story #2 (Scenario #1): Secure Device Authentication	180
10.7.3	User Story #3 (Scenario #2): Secure transmission of patient data	182
10.7.4	Metrics of Success	186
11	PiQASO UC6.1 - Sector: Aerospace - Integration	188
11.1	Motivation and Challenges in Aerospace HIL Testing	188
11.1.1	Current Landscape: Hardware-in-the-Loop Validation in Aviation	188
11.1.2	The Convergence of Cybersecurity and Flight Safety	188
11.1.3	The Transition from Classical to Post-Quantum Cryptography in Aerospace	189
11.1.4	Motivation for PiQASO's Role in Aerospace Testing	189
11.2	Use Case Description, Goals, and Objectives	190
11.3	PiQASO Functionalities	191
11.3.1	Rationale for the Phased Implementation	191
11.4	"As-is" Scenario	192
11.5	Components and Roles Considered	193

11.5.1 System Architecture: Before vs After PQC	193
11.5.2 Virtual Testing – Sim-in-the-Loop	195
11.5.3 Components, Entities and Roles	196
11.6 Challenges and Needs from PiQASO	196
11.7 “To-Be” Reference Scenario #1: Quantum-Safe Real-Time Control	198
11.8 “To-Be” Reference Scenario #2: Secure Distributed HIL with IP Protection	198
11.9 “To-Be” Reference Scenario #3: Hardware-Accelerated HIL Control	198
11.10 Data Model and Transport Pattern	199
11.11 Reference Scenario User Stories and Metrics of Success (KPIs)	199
11.11.1 User Story #1 (Scenario #1): Secure Control Signal Transmission	199
11.11.2 User Story #2 (Scenario #2): Protected IP & Telemetry Logs (ABE = role-based / RBAC)	202
11.11.3 User Story #3 (Scenario #3): Bounding HW Acceleration in the HIL Loop	203
11.11.4 Application-Specific KPIs	204
12 PiQASO UC6.2 - Sector: Aerospace - Quantum-Secure Aviation Maintenance Services Email Communication	206
12.1 Use Case’s Description, Goals and Objectives	206
12.2 “As-Is” Scenario	207
12.3 Components and roles considered for UC6.2	209
12.4 Challenges and Needs from PiQASO	210
12.5 “To-Be” Reference Scenario #1: Minimal-Intrusion Integration (Internal Communication)	212
12.6 “To-Be” Reference Scenario #2: Extended-Message Integration (External Communication)	213
12.7 Reference Scenario User Stories and Metrics of Success	215
12.7.1 User Story #1 (Reference Scenario #1): Secure Internal Email Exchange	215
12.7.2 User Story #2 (Reference Scenario #1): Secure External Email Exchange	218
12.7.3 Key Performance Indicators	221
13 PiQASO UC7 – Sector: Online Media	222
13.1 Use Case’s Description, Goals and Objectives	223
13.2 PiQASO Functionalities	224
13.3 Components and Roles Considered	224
13.4 “As-Is” Scenario	225
13.5 Challenges and Needs from PiQASO	226
13.6 “To-Be” Reference Scenario #1: Quantum-Safe Authentication and CMS Access	227
13.7 “To-Be” Reference Scenario #2: Privacy-Preserving Article Sharing with Updatable (Hybrid) Public-Key Encryption	228
13.8 Reference Scenario User Stories and Metrics of Success	229
13.8.1 User Story #1 (Reference Scenario #1): Quantum-Safe Authentication and CMS Access (Journalist / Editor)	229
13.8.2 User Story #2 (Reference Scenario #1): Privacy-Preserving Article Upload and Review (UHE/UPKE)	231
13.8.3 Key Performance Indicators	233

14 PiQASO UC8 – Sector: UAVs	236
14.1 Use case's Description, Goals and Objectives	236
14.2 PiQASO Functionalities	237
14.3 “As-is” Scenario	237
14.4 Components and roles considered in the context of UAV Use Case	240
14.4.1 System model and common communication path	240
14.4.2 Mapping to the reference scenarios	241
14.4.3 Assumptions	241
14.4.4 Scope of ABE	242
14.4.5 Common cryptographic backend interface	242
14.4.6 Cryptographic Execution Paths and End-to-End Protection	243
14.5 Challenges and Needs from PIQASO	244
14.5.1 Security needs during UAV operations	244
14.5.2 Threat Landscape for UAV and Ground Station Communications	245
14.5.3 Vulnerabilities across the UAV mission lifecycle	247
14.5.4 How the PiQASO Framework Addresses These Security Challenges	249
14.6 “To-Be” Reference Scenario #1: On-device PQC via PQC SDK and PQC-TCB/QR-TCB	251
14.7 “To-Be” Reference Scenario #2: PQCaaS-assisted PQC via PQ-DS for multi-ground and swarm scalability	252
14.8 Reference scenario user stories and Metrics of success	253
14.8.1 User Story #1 (Reference Scenario #1): Secure pre-flight channel establishment as an arming gate	254
14.8.2 User Story #2 (Reference Scenario #1): Secure mission upload and continuous C2 protection	254
14.8.3 User Story #3 (Reference Scenario #1): PQCaaS-enabled multi-ground and swarm governance	256
14.8.4 Testing and validation approach (UC8 evaluation plan)	258
14.8.5 Deployment plan (integration and demonstration sequencing)	258
14.9 Metrics of Success	259
15 PiQASO UC9 – Sector: Transportation	260
15.1 Use case's Description, Goals and Objectives	260
15.2 PiQASO Functionalities	261
15.3 “As-is” Scenario	261
15.4 Components and Roles Considered	262
15.5 Challenges and Needs from PiQASO	263
15.6 “To-Be” Reference Scenario #1: Quantum-Safe Mission Configuration	264
15.7 “To-Be” Reference Scenario #2: Secure Data Storage and Retrieval in the Cloud	265
15.8 Reference scenario user stories and Metrics of success	265
15.8.1 User Story #1 (Reference Scenario #1): Quantum-Safe Mission Upload (Operator)	265
15.8.2 User Story #2 (Reference Scenario #1): Authentic Anomaly Reporting (OBU)	267
15.8.3 User Story #3 (Reference Scenario #1): Secure Anomaly Retrieval (FCG Maintenance Engineer)	268

16 PiQASO PQC Artifacts Testing Plan per Use Case	270
17 Summary and Conclusions	278
Bibliography	280

List of Figures

1.1	Relation of D2.1 to other PiQASO WPs and Deliverables	23
2.1	QR-TCB Architecture model.	30
2.2	System model representation with registration, data upload and data retrieval functionalities	36
3.1	PiQASO PQ-TLS - Variant 1: Direct Client-Server Authentication	56
3.2	PiQASO PQ-TLS - Variant 2: Authentication through PQCaaS	58
3.3	PiQASO Updatable Public Key Encryption	60
3.4	PiQASO Attribute-Based Encryption (ABE)	62
5.1	Automotive vPKI Architecture	80
5.2	Automotive vPKI Sequence Diagram	81
5.3	Automotive AS-IS System Architecture	83
5.4	Automotive AS-IS vPKI Architecture	84
5.5	Automotive AS-IS Sequence Diagram	86
5.6	Automotive AS-IS vPKI Certificate Provisioning Sequence Diagram	88
5.7	Automotive TO-BE Key Management	91
5.8	Automotive TO-BE System Architecture	92
5.9	Automotive TO-BE Sequence Diagram - Certificate Provisioning and Validation	94
5.10	Automotive TO-BE Sequence Diagram - Auth, Secure Key and V2X Message Communication	98
5.11	Automotive TO-BE Sequence Diagram - Quantum-Secure OTA Update	102
6.1	As-is AMR fleet communication architecture (VPN overlay with unprotected ROS 2 traffic).	113
6.2	Logical component diagram of the Fleet Manager, AMRs, middleware, and operators. . . .	114
6.3	To-be Scenario #1: Secured ROS 2/DDS communication for navigation and task allocation.	119
6.4	To-be Scenario #2: Secured Fleet Manager with selective sharing and protected storage. .	122
6.5	To-be Scenario #3: ASSIST – Secure Remote Assistance with AKE-based communica- tions and hardware-verified manual control.	123
7.1	Financial e-banking architecture and workflow.	126
7.2	RSA Key distribution in actual online banking.	127
7.3	PQC Protection for External Communications via PiQASO.	129

7.4	PiQASO SDK used as a PQaaS.	131
7.5	FPGA Hardware Acceleration Evaluation Testing.	133
8.1	UC4 "As-Is" Smart Grid Communication Architecture	138
8.2	US1: Secure Grid Telemetry	143
8.3	US2: Crypto-Agile Algorithm Transition	144
8.4	US3: Quantum-Secure Edge Device Onboarding	146
9.1	TelluCare Remote Patient Monitoring architecture and workflow	150
9.2	The Management (right)- and Administration (left) environments of the PHG	151
9.3	Tellu PHG - IOT-GATE-IMX8	151
9.4	AS_IS onboarding of PHG to Tellu cloud sequence diagram	154
9.5	AS_IS sequence diagram for data transfer from PHG to Tellu services	155
9.6	AS_IS process for the update of the gateway sequence diagram	157
9.7	TO_BE onboarding of PHG to Tellu cloud sequence diagram	162
9.8	TO_BE sequence diagram for data transfer from PHG to Tellu services	164
9.9	TO_BE sequence diagram for the update of gateways	167
10.1	Smart Ambulance Scenario: Overall View	171
10.2	Ambulance UC Architecture: Components	173
10.3	"as-is" Smart Ambulance: GW Authentication	175
10.4	"as-is" Smart Ambulance: User Authentication Steps	176
10.5	"As-is" Smart Ambulance: Sending patient data	177
10.6	User Story 1: User Authentication with PQC-TLS	179
10.7	User Story 2: GW AUthentication with PQC-TLS	181
10.8	User Story 3: Secure transmission of patient data	183
10.9	User Story 3.1: Querying on patient data with ABE or Functional Encryption	185
11.1	Operation process of the HIL rig before PiQASO: one control cycle with command and feedback exchanged in plaintext over the MQTT broker.	192
11.2	Demonstrator architecture before PiQASO: command and feedback cross the broker in plaintext.	194
11.3	Demonstrator architecture after PiQASO: a pq_frame daemon at each endpoint secures the payloads; the broker relays ciphertext only.	195
11.4	HIL operation after PiQASO: one secured control cycle (PQC mode) – command and feedback are sealed end-to-end with the AES-256-GCM session key, and the broker relays ciphertext only.	197
11.5	Mutually authenticated post-quantum key agreement: an ML-DSA-signed, ephemeral ML-KEM exchange yields the AES-256-GCM session key. The encapsulation ciphertext is the only key material on the wire; the shared secret is derived independently at each end.	201

11.6 ABE archival and selective decryption: test logs are sealed under a role-based attribute policy; only a holder of the matching role attributes can decrypt.	203
12.1 Functional description of email client-server communication in the “As-Is” scenario.	208
12.2 High-Level Architecture of the QS-AMEC Integration, illustrating internal PQC boundaries and the Internal Email Server acting as an Edge Decrypter.	209
12.3 Sequence diagram for PQC-secured email exchange - Mode A (Local SDK)	216
12.4 Sequence diagram for PQC-secured email exchange - Mode B (Central KMS)	217
12.5 Sequence diagram for PQC-secured 1-to-many email exchange (Edge Decryption)	220
13.1 Article Sharing and Storage Workflow	226
13.2 Sequence diagram for Quantum-Safe Authentication and CMS Access	230
13.3 Sequence diagram for Privacy-Preserving Article Upload and Review (UHE/UPKE)	232
13.4 Sequence diagram for Privacy-Preserving Article Upload and Review (ABE)	233
14.1 Figure 1 High level illustration of the use case 8	239
14.2 Figure 2 UAV as is high level architecture	239
14.3 Figure 2 Piqaso High Level UAV Architecture	240
14.4 UC8 Sequence Diagram using onboard secure node	255
14.5 UC8 Sequence Diagram using PQCaaS	257
15.1 Architecture of the “as-is” scenario.	262
15.2 Sequence diagram of the “as-is” scenario.	263
15.3 Sequence diagram of User Story #1.	266
15.4 Sequence diagram of User Story #2.	267
15.5 Sequence diagram of User Story #3.	269

List of Tables

2.1	Summary of PiQASO Framework features	25
3.1	Common evaluation metrics for all PiQASO use cases	48
3.2	Acceptance criteria for the common evaluation metrics in the Automotive use case (UC1) .	49
3.3	Acceptance criteria for the common evaluation metrics in the Automation use case (UC2) .	49
3.4	Acceptance criteria for the common evaluation metrics in the Finance use case (UC3) . . .	50
3.5	Acceptance criteria for the common evaluation metrics in the Energy use case (UC4) . . .	50
3.6	Acceptance criteria for the common evaluation metrics in the Healthcare use case (UC5.1)	51
3.7	Acceptance criteria for the common evaluation metrics in the Healthcare use case (UC5.2)	52
3.8	Acceptance criteria for the common evaluation metrics in the Aerospace use case (UC6.1)	53
3.9	Acceptance criteria for the common evaluation metrics in the Aerospace use case (UC6.2)	53
3.10	Acceptance criteria for the common evaluation metrics in the Online Media use case (UC7)	53
3.11	Acceptance criteria for the common evaluation metrics in the UAVs use case (UC8)	54
3.12	Acceptance criteria for the common evaluation metrics in the Transportation use case (UC9)	55
4.1	CAI analysis modes and architectural role	65
4.2	UC1 (ABI) Sector: Automotive — cryptographic asset inventory	69
4.3	UC2 (PAL) Sector: Automation — cryptographic asset inventory	70
4.4	UC3 (CXB) Sector: Finance: Secure Online Payments (e-wallet, e-banking) — crypto- graphic asset inventory	70
4.5	UC4 (MOH-MORE-CERTH) Sector: Energy — cryptographic asset inventory	70
4.6	UC5.1 (TELLU) Sector: Healthcare — cryptographic asset inventory	71
4.7	UC5.2 (PARTICLE) Sector: Healthcare — cryptographic asset inventory	71
4.8	UC6.1 (BIBA) Sector: Aerospace Integration — cryptographic asset inventory	72
4.9	UC6.2 (NEDHO) Sector: Aerospace Integration — cryptographic asset inventory	72
4.10	UC7 (DPG) Sector: Online Media — cryptographic asset inventory	72
4.11	UC8 (ACCELI) Sector: UAVs — cryptographic asset inventory	73
4.12	UC9 (STROWL) Sector: Transportation — cryptographic asset inventory	74
5.1	PiQASO Functionalities to be evaluated for Use Case 1 (Automotive)	79
5.2	Performance and Security KPIs for the Automotive Use Case	106

6.1	PiQASO Functionalities to be evaluated for Use Case 2 (Automation)	111
6.2	KPIs for Use Case 2 (Automation)	121
7.1	PiQASO Functionalities to be evaluated for Use Case 3 (Finance)	125
7.2	Consolidated UC3 KPIs for Security, Real-Time Performance, Scalability and Usability	134
8.1	PiQASO Functionalities to be evaluated for Use Case 4 (Energy)	136
8.2	Consolidated Key Performance Indicators (KPIs) for the Energy Use Case (UC4)	147
9.1	Medical device models	152
9.2	PiQASO Functionalities to be evaluated for eHealth usecase	153
9.3	PiQASO Application-Specific KPIs for the TelluCare Remote Patient Monitoring Use Case	168
10.1	PiQASO Functionalities to be evaluated for the Smart Ambulance UC	172
10.2	PiQASO KPIs for the Smart Ambulance UC	186
11.1	PiQASO Functionalities to be evaluated for UC6.1 (Aerospace Integration)	191
11.2	Real-time payloads protected in UC6.1.	199
11.3	Application-specific KPIs for UC6.1 (Aerospace Integration).	204
12.1	PiQASO Functionality Rollout Roadmap for QS-AMEC	211
12.2	Application-specific KPIs for Secure Email Exchange Use Case	221
13.1	PiQASO Functionalities to be evaluated for Use Case 7 (Online Media)	224
13.2	Application-specific KPIs	233
14.1	PiQASO Functionalities to be evaluated for Use Case 8 (UAVs)	237
14.2	UC8 mapping to the reference scenarios	241
14.3	UC8 cryptographic execution paths and end-to-end protection	243
14.4	Application-Specific Key Performance Indicators (KPIs) for the UAV Use Case	259
15.1	PiQASO Functionalities to be evaluated for Use Case 9 (Transportation)	261
15.2	KPIs for evaluating UC9 success across security, performance, reliability, and usability. Concrete values for X, Y and SLAs will be determined during implementation and trials.	269
16.1	Functionalities for the Automotive use case	270
16.2	Functionalities for the Automation use case	271
16.3	Functionalities for the Finance use case	271
16.4	Functionalities for the Energy use case	272
16.5	Functionalities for the Healthcare use case	273
16.6	Functionalities for the Aerospace use case	273
16.7	Functionalities for the Online Media use case	274

16.8 Functionalities for the UAV use case	275
16.9 Functionalities for the Transportation use case	276

Chapter 1

Introduction

Cryptographic services constitute the backbone of trust in modern service graph chain infrastructures, enabling secure communications, authenticated command and control, controlled information sharing, and trustworthy platform operation across heterogeneous systems ranging from embedded endpoints to backend infrastructures. However, the advent of quantum computing challenges the computational hardness assumptions of traditional cryptographic schemes, thus compromising the security level of existing cryptographic systems and **Public Key Cryptography (PKI)** infrastructures. In this regard, PiQASO aims to quantum-harden a wide variety of service graph chains belonging to diverse modern application domains through the application of **Post-Quantum Cryptography (PQC)**, which may be characterized by long platform lifecycles, constrained and intermittently connected tactical nodes, stringent assurance expectations, and adversarial capabilities that extend beyond remote cyber attacks to include physical access, side-channel exploitation, and fault injection.

In this regard, the goal of D2.1 is to first establish a concrete overview of the vision and background of PiQASO, detailing existing work that has been performed in the domain of future-proof, quantum-resistant cryptography, and identifying the actions that need to be taken by PiQASO towards achieving the goal of post-quantum transition of modern service graph chains. This is in continuation to D3.1 [55], where the necessity of PQC migration was framed and positioned as a system-level transformation affecting protocols, PKI ecosystems, and deployability constraints. In D3.1, we also provided the technical foundation for the work to be performed throughout the project lifecycle, by describing not only the mathematical principles behind the functionalities of PiQASO, but also the detailed overview of all provided functionalities, namely **Forward Security** through **Updatable Public Key Encryption (UPKE)**, **Functional Encryption**, decompressing **Learning-with-Errors (LWE)** with Polynomial Matrices, and the **Algorithmic Optimization and HW-based Acceleration** capabilities of PiQASO. In this deliverable, we expand upon this by providing a definition of the PiQASO architecture (Chapter 2), which details the component positioning and interactions in a structured manner that enables the post-quantum transition of the target infrastructures. In Chapter 4, particular focus is placed on the **Crypto Asset Inventorying (CAI)** component, whose purpose is to build a structured, evidence-based understanding of the cryptographic and software-dependency posture of an operational environment before any post-quantum protection mechanisms is introduced. In this regard, we also provide detailed results of the application of the CAI in the context of the PiQASO use case demonstrators.

The PiQASO architecture is characterized by a set of requirements, which need to be fulfilled for achieving the aforementioned post-quantum transition goals in a wide range of complex, large-scale, heterogeneous service graph chains. To this end, in Chapter 3, we define the PiQASO methodology for the elicitation of these requirements, beginning with the design of the PiQASO Minimum Viable Product (MVP), which serves to demonstrate the common vision of the consortium, and has been carefully planned to incorporate the activities scheduled for the project demonstrators. This systematic approach ensures a

coherent and well-structured foundation for the subsequent stages of the PiQASO project. We proceed with the requirements elicitation methodology adopted in PiQASO, which follows a top-down approach to ensure that the framework requirements are grounded in operational reality and remain reusable across heterogeneous missions and infrastructures.

The functionalities of PiQASO culminate into the provision of the **PiQASO PQC Ensemble**, offering the functionalities and primitives to be applied in the context of the use cases, which are then going to be tested through the design and implementation of an appropriate testbed. These use cases represent a wide variety of service graph chains and demonstrate the applicability of PiQASO in the context of diverse application domains, including (i) **Automotive** (Chapter 5), (ii) **Automation** (Chapter 6), (iii) **Finance** (Chapter 7), (iv) **Energy** (Chapter 8), (v) **Healthcare** (Chapter 9), (vi) **Aerospace** (Chapter 11), (vii) **Online Media** (Chapter 13), (viii) **UAVs** (Chapter 14), and (ix) **Transportation** (Chapter 15). Thus, *a core aim of this deliverable is to provide a detailed description for the post-quantum transition of all envisioned use case demonstrators of PiQASO in their respective Chapters, as well as the needs that need to be fulfilled by PiQASO, based on the goals and objectives identified in the context of each use case.*

The first core step of this process is *mapping the PiQASO functionalities which are the most appropriate in the context of each use case for achieving its goal of post-quantum transition*. This is achieved through the application of the aforementioned **Crypto Asset Inventorying (CAI)** tool, which identifies the most crucial assets and cryptographic operations that should be prioritized for quantum hardening. The application of the CAI in each use case, as well as the output of this process, is detailed in Chapter 4. The basic purpose of this process is to identify both which are the *common requirements to accommodate post-quantum migration regardless of the application domain*, and the *set of targeted security requirements associated to specific properties in the context of the use cases*. For instance, as will be shown in Chapter 4, a common characteristic of all use cases is the use of traditional **TLS** for authentication, which needs to be elevated to **Post-Quantum TLS (PQ-TLS)**. However, depending on specific characteristics of each use case, the acceptance criteria (e.g., latency, bandwidth, computational overhead) may differ depending on the application domain, and we need to identify the acceleration and optimization of PQC primitives that lead to their fulfilment. Application-specific requirements may be related to the safety criticality and sensitivity of data, which leads to the need for advanced high-end schemes, such as **Functional Encryption** and **Attribute-Based Encryption (ABE)**.

Then, for evaluating the performance of the PiQASO offerings selected as output of this process, for each use case we identify a set of **Reference Scenarios**, as well as **User Stories** which define a concrete action workflow that will guide the experimentation activities in each use case through the PiQASO testbed. Based on this, we identify all PiQASO functionalities that need to be integrated in the context of all use cases. In Chapter 16, we provide tables detailing the functionalities selected as needed for the post-quantum transition of each corresponding application domain, the description of their usage in the context of each use case, and the expected implementation time per use case.

Overall, this deliverable provides a comprehensive overview of the PiQASO architecture, and details its functionality in the context of a wide range of practical applications and use case demonstrators. This deliverable provides the basis for the evaluation of PiQASO in the context of these demonstrators, and defines the user stories that will guide the experimentation activities of PiQASO.

1.1 Relation to other WPs and Deliverables

As previously outlined, one core target of D2.1 is to provide the PiQASO reference architecture, consisting of a structured representation of the required security functionalities for the safety-critical services of the

PiQASO use cases, to define a layered architecture to enable the transition of legacy systems to next-generation quantum-secure systems. This reference architecture is grounded in the initial definition of PiQASO components and the migration plan established in D3.1, which provides the basis for refining the PiQASO architecture and selecting the PiQASO components to be used for fulfilling all use case requirements. D2.1 also provides a more concrete version of the threat model initially defined in D3.1, and defines the long-term security requirements and functional specifications for the PiQASO use cases.

D2.1 provides input to other PiQASO WPs, and sets the foundation for the integration and evaluation activities to be carried out in the context of all use case demonstrators. Most directly, D2.1 feeds into D4.1, where the reference scenarios and user stories defined in this deliverable provide a concrete evaluation plan that will guide the experimentation activities for all use case demonstrators, and will be used for determining their success based on defined acceptance criteria and KPIs. D2.1 also provides the basis for the integration activities of the PiQASO PQC Ensemble across all use case demonstrators in WP4, thus setting the scene for the evaluation of all PiQASO modalities and the assessment of their impact on the normal operational behaviour of the use case environments. The outputs of D2.1 further provide input to WP5, where the dissemination of PQC knowledge and the training of all concerned stakeholders will be facilitated through the construction of the PiQASO Academy. Finally, in the context of WP6, dissemination activities will be carried out for facilitating the adoption of all PiQASO offerings in actual, real-world organizations, and detailed exploitation plans for PiQASO artefacts will be set forth.

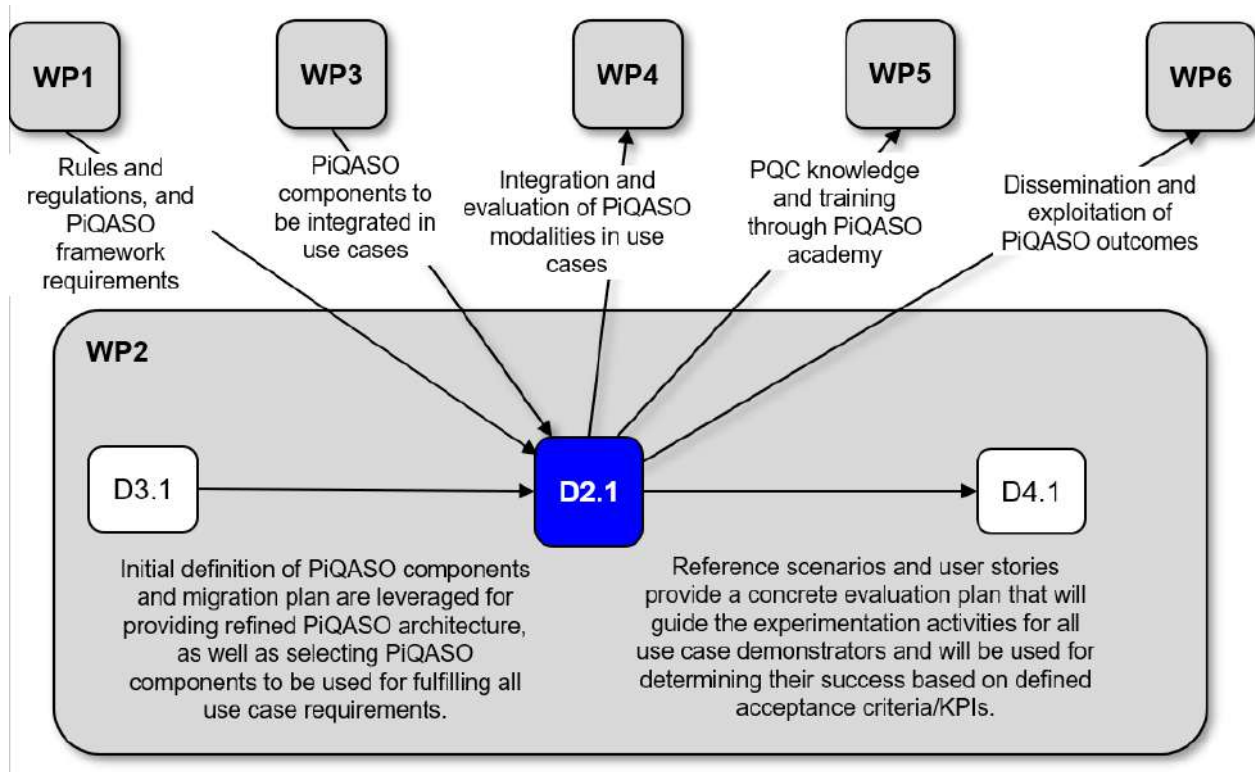


Figure 1.1: Relation of D2.1 to other PiQASO WPs and Deliverables

1.2 Deliverable Structure

The remainder of this deliverable is structured as follows:

Chapter 2 provides a refined overview of the PiQASO architecture, and describes core PiQASO functionalities in a more refined and detailed manner compared to D3.1.

Chapter 3 details the experimentation methodology of PiQASO, entailing the definition of the Minimum Viable Product (MVP), requirements elicitation, and Key Performance Indicators (KPIs).

Chapter 4 details the Crypto Asset Inventorying (CAI) component, and provides results from the application of the CAI in the context of the PiQASO use cases.

Chapter 5 provides the description, architecture, reference scenarios, and user stories for the Automotive use case.

Chapter 6 provides the description, architecture, reference scenarios, and user stories for the Automation use case.

Chapter 7 provides the description, architecture, reference scenarios, and user stories for the Finance use case.

Chapter 8 provides the description, architecture, reference scenarios, and user stories for the Energy use case.

Chapter 9 provides the description, architecture, reference scenarios, and user stories for the Healthcare use case.

Chapter 11 provides the description, architecture, reference scenarios, and user stories for the Aerospace use case.

Chapter 13 provides the description, architecture, reference scenarios, and user stories for the Online Media use case.

Chapter 14 provides the description, architecture, reference scenarios, and user stories for the UAVs use case.

Chapter 15 provides the description, architecture, reference scenarios, and user stories for the Transportation use case.

Chapter 16 provides a comprehensive overview of all PiQASO functionalities employed in the context of all envisioned use case demonstrators, in order to fulfil all their needs and challenges.

Chapter 17 concludes the deliverable.

Chapter 2

PiQASO Conceptual Architecture and Functional Components

2.1 Summary of PiQASO Conceptual Architecture

In D3.1 [55], we provided a detailed roadmap towards achieving the core target of PiQASO of *enabling the post-quantum transition of modern infrastructures and complex mixed-criticality service graph chains, even those with legacy architectures and employing resource-constrained devices with limited capabilities for executing complex mathematical cryptographic operations*. To this end, we provided a description of the main building blocks of the PiQASO Framework, and their role towards achieving this target. Here, we expand upon this definition, through a more detailed description of the placement of these components into a structured approach, targeted towards a holistic post-quantum transition strategy for mixed-criticality service graph chains. This will then provide the basis for the integration of the PiQASO Framework into all use case demonstrators, and will eventually facilitate the formulation of the scenarios and user stories that will guide the evaluation activities of PiQASO. In Table 2.1, we provide a high-level description of all components comprising the PiQASO architecture.

Table 2.1: Summary of PiQASO Framework features

PiQASO Component	Description
Crypto Asset Inventorying (CAI) (Chapter 4)	The CAI acts as an entry point to the post-quantum transition pipeline of PiQASO, by providing a structured and evidence-based understanding of the cryptographic posture of an operational environment, considering all underlying software dependencies, prior to the introduction of PQC mechanisms. To this end, the CAI component scans system components to identify cryptographic algorithms, protocols, and dependencies in use. As a result, it identifies which assets, communication flows, and services are most critical and should be prioritised for post-quantum protection. The Software Bill Of Materials (SBOM) and Cryptographic Bill Of Materials (CBOM) are orthogonal to the CAI, as they are generated after its execution, and contain information regarding which are the vulnerable endpoints of the system which need to be secured against quantum threats. Specifically, the CBOM is a structured representation of all cryptographic assets within the system and supports vulnerability assessment and migration planning, by capturing cryptographic objects and operational relationships, connected to the software, service, or platform context where these objects are instantiated. Conversely, the SBOM explains the dependencies of software components between them, or software packages and libraries, without necessarily explaining how cryptography is used by those components.

PQC SDK (Section 2.2)	The PQC SDK is dedicated to high-quality, embedded-oriented implementations of post-quantum cryptographic (PQC) algorithms, and provides post-quantum cryptographic mechanisms for securing data at the edge, including encryption, authentication, and digital signatures. It focuses on providing lightweight asymmetric cryptographic primitives suitable for resource-constrained platforms, including also countermeasures against the threat model of PiQASO defined in D3.1 [55]. Specifically, the implemented algorithms include ML-KEM (key encapsulation), ML-DSA (digital signatures), and SLH-DSA (digital signatures), and countermeasures against side-channel attacks include first- and second-order masking and shuffling techniques.
QR-Trusted Computing Base (QR-TCB) (Section 2.3)	The Quantum-Resistant Trusted Computing Base (QR-TCB) considered in PiQASO is designed as the minimal trusted SW stack to support the execution of the PQC SDK, responsible for protecting long-term cryptographic assets and enforcing secure execution of PQC, and the architecture of the QR-TCB is based on the core security principle that long-term secret key material must remain exclusively within the TEE boundary, introducing variations for short-term secrets depending on the desired security models. The core consideration of the QR-TCB is to find the optimal breakdown of the security operations that need to be executed in isolation, so as to not compromise the security profile of the device, while also achieving high performance. To this end, the QR-TCB supports two core configurations, namely: (i) Minimal TEE , where the decapsulation procedure runs inside the TEE while the encapsulation procedure runs in the untrusted environment, thus minimizing TEE resource usage and latency, and (ii) Full TEE , where both encapsulation and decapsulation run within the TEE, thus achieving stronger security guarantees on the encryption process and ensures complete protection of both key material and plaintext data.
Algorithmic Optimization and HW-based Acceleration (Section 2.4)	The algorithmic optimizations and HW-based acceleration capabilities of PiQASO aim to address performance bottlenecks that have been identified in PQC algorithms, such as ML-KEM, ML-DSA, SLH-DSA, and HQC. These may include mathematical computations such as the Number Theoretic Transform (NTT) for lattice-based schemes, the SHA-3 and SHAKE hash functions, and the high-degree polynomial multiplication in code-based crypto. In addition, the integration of countermeasures against Side-Channel Attacks (SCA) also increases execution time and memory footprints. To address these issues, the approach followed by PiQASO is threefold: (i) Acceleration of cryptographic hash primitives through a dedicated hardware core for supporting first-order countermeasures natively, (ii) acceleration of polynomial multiplication through a parallelized architecture equipped with multiple butterfly units for reducing latency in lattice-based PQC, and (iii) integration of the proposed post-quantum accelerators into a high-performance software/hardware co-design architecture.
PQC-enabled High-level Protocol Integration (Section 2.2)	Post-quantum cryptographic primitives included in the PQC SDK (e.g., ML-KEM and ML-DSA) are also integrated into higher-level protocols in order to quantum harden protocols used for securing network communications in mixed criticality service graph chains, and particular focus is placed on the post-quantum transition of the TLS protocol for establishing secure and authenticated communication channels, i.e., PQ-TLS. In TLS 1.3, ML-KEM is being adopted as a key exchange mechanism, typically in hybrid configurations that pair it with classical ECDH, ensuring security even if one primitive is broken. ML-DSA can also replace ECDSA or RSA for certificate signatures and handshake authentication. PiQASO may also investigate the possibility of quantum-hardening the IPsec protocol, by using ML-KEM to replace the existing key exchange framework and ML-DSA to handle peer authentication.

PQC-as-a-Service (PQCaaS) (Section 2.5)	Provides externalised post-quantum cryptographic capabilities for resource-constrained devices, as well as cloud-based security services. The PQCaaS component is intended to provide devices that do not possess the computational resources to execute complex cryptographic operations with the capability to run the PQC schemes offered by the PiQASO Framework, and enables devices to perform secure storage of their own data, i.e., perform encryption at rest. Devices with low computational power typically do not possess the capability to create their own keys. Thus, For cloud-based data secrecy, the PQCaaS offers the capability to construct a new key protected through application data workloads, and a PQ Data Encryption Server is responsible for performing the encryption of the data, so that devices do not need to perform the encryption themselves. This enables the implementation of policy-based Updatable Public Key Encryption (UPKE), leveraging post-quantum cryptographic primitives, so that the encryption server can encrypt a set of data at rest under a specified access policy, with cryptographic keys updated at regular epochs (i.e., predefined time intervals). Such a cryptographic mechanism is particularly beneficial in scenarios where the encryption server cannot be assumed trusted, or where stronger assurances for security, privacy, and unlinkability are needed. In this case, the PiQASO PQCaaS solution provides the first-of-its-kind lattice-based construction for UPKE, which will be documented in D3.2. In this regard, PiQASO aims to design and implement the additional modality of policy-based UPKE from lattice assumptions in order to mediate the secure and privacy-preserving data sharing and storage services.
QR-Ledger	The QR-Ledger of PiQASO is used for auditability purposes, and ensures immutable and verifiable logging of events and metadata using post-quantum secure mechanisms. It can be used in order to achieve tamper-proof storage of metadata in functional-oriented operations, so that they can be provided in a secure and auditable manner in the context of the PiQASO use cases. In this regard, the PiQASO plan for the QR-Ledger is threefold: (i) First, hash-based signatures such as SLH-DSA, WOTS, and WOTS+ will be leveraged for quantum-resistant on-chain storage of data in the QR-Ledger. The first version of the QR-Ledger provided by PiQASO will be tailored to evaluate the use of such hash-based signatures for the storage of on-chain data. (ii) The next version will extend the on-chain storage capabilities with first-of-its-kind on-chain verification, by employing <i>pre-defined smart contract templates</i> so that authenticated parties are able to verify hash-based signatures for data stored on-chain. (iii) Finally, PiQASO envisions an on-chain validation scheme, which entails quantum-hardening the validation process. This is typically based on Elliptic Curve Crypto (ECC) and endorses the correctness of a transaction through a majority-based scheme with multiple validators. Note that the validation scheme of PiQASO only refers to the post-quantum extension of the Ethereum technology, and does not apply to other types of chains (e.g., Besu).

Thus, considering the components offered by PiQASO as detailed in Table 2.1, the high-level action workflow of the PiQASO architecture can be summarized as follows: First, the **Crypto Asset Inventorying** tool is responsible for analysing the architecture of the environment that needs to be quantum secured, and identifying the components which are more critical and need to be quantum hardened. This information is contained in the **Cryptographic Bill of Materials**, which details the vulnerable endpoints that need to be secured against quantum threats for guiding the migration planning process. Then, the **PQC SDK** provides the required PQC algorithms for supporting this process, including the appropriate low-level PQC algorithms (e.g., ML-KEM, ML-DSA), as well as the high-level protocols which may be used by the target infrastructure (e.g., PQ-TLS, IPsec). For devices with the computational capability to perform the required mathematical operations, the **QR-TCB** enables the devices to handle the cryptographic material and keys for supporting the execution of PQC. For devices which do not possess the required computational capabilities, the **PQCaaS component** supports offloading the execution of PQC to a dedicated server. Finally, information such as trust evidence and audit logs (e.g., attestation outcomes, policy decisions) stemming from the execution of PQC are stored in the **QR-Ledger**, where they can be made

accessible only to entities which can verifiably demonstrate the possession of the appropriate attributes through the Attribute-Based Encryption **ABE** and **UPKE** schemes.

2.2 PiQASO PQC Software Development Kit (SDK)

The PiQASO PQC SDK is a C-based software library for post-quantum cryptography (PQC), specifically designed for embedded applications. It requires no external dependencies and supports two NIST-standardized PQC schemes as modular components: ML-KEM (FIPS 203) and ML-DSA (FIPS 204). Users can embed all three security levels of each scheme simultaneously. One of the PQC SDK's key features is its ability to handle multiple security levels for the same scheme, with configurable options set at compile time:

- To modify embedded schemes, edit `CMakeConfig.cmake` and set the relevant option to ON. By default, the PQC SDK includes both ML-DSA and ML-KEM.
- To adjust available security levels, directly edit the module configuration files (`pqc_sdk_mldsa_config.h` and `pqc_sdk_mlkem_config.h`).

The PQC SDK relies on static memory allocation for cryptographic operations. The required memory size is determined at compilation time based on the selected schemes and security levels. The cryptographic algorithms embedded by the PQC SDK are constant-time.

2.2.1 PQC SDK Interface

The PQC SDK's user interface is straightforward and centers around "contexts", which define a scheme's configuration and security level. The library provides two context types:

- `pqc_sdk_mldsa_mem_ctx_t` for the ML-DSA scheme.
- `pqc_sdk_mlkem_mem_ctx_t` for the ML-KEM scheme.

Each context contains all scheme-specific parameters for a given security level, along with the PQC SDK's memory mapping for cryptographic operations. Users can define multiple contexts for different schemes or security levels, but all operations share the same memory space. Before using a context, you shall initialize it with the appropriate function (`pqc_sdk_mldsa_init` or `pqc_sdk_mlkem_init`). This function takes the context and the desired security level as inputs and prepares it for cryptographic computations.

The following functions are available for ML-DSA:

- `pqc_sdk_mldsa_keygen`: Generates a public/private key pair. This function takes as input the output buffers for the public and private keys, as well as the previously initialized context for a given security level.
- `pqc_sdk_mldsa_sign`: Generates a signature for a message. This function takes as input the buffer for the generated signature and a variable that will store the signature's length. Input data includes the private key, the message to be signed and its length, additional data to be included in the signature and its length, and the previously initialized context for a given security level.

- `pqc_sdk_mldsa_verif`: Verifies a message's signature. This function takes as input the signature and its length, the public key, the message and its length, additional data and its size, and the previously initialized context for a given security level. The function returns `pqc_sdk_SUCCESS` if the signature is valid; otherwise, it returns a different error code.

The following functions are available for ML-KEM:

- `pqc_sdk_mlkem_keygen`: Generates an encapsulation/decapsulation key pair. This function takes as input the output buffers for the encapsulation and decapsulation keys, as well as the previously initialized context for a given security level.
- `pqc_sdk_mlkem_encaps`: Encapsulates a shared secret and returns it. This function takes as input the buffer that will store the shared secret and the ciphertext. Input data includes the encapsulation key and its size, as well as the previously initialized context for a given security level.
- `pqc_sdk_mlkem_decaps`: Decapsulates a shared secret from a ciphertext and a decapsulation key. It takes as input the buffer for the shared secret and, as input data, the decapsulation key and its size, the ciphertext and its length, and the previously initialized context for a given security level.

All functions return an error code, which should be `pqc_sdk_SUCCESS` if the cryptographic computation completes without errors. If a different code is returned, an error has occurred, possibly due to malformed inputs (or an invalid signature in the case of ML-DSA verification). After each operation, the PQC SDK automatically clears its internal memory to prevent sensitive data leaks, but the context remains active for subsequent operations without re-initialization.

2.2.2 Functional tests

The PQC SDK has been rigorously tested throughout development to ensure reliability. Tests covered both user-facing and internal functions, using two reference implementations as golden references: PQClean¹ and `mldsa-native/mlkem-native`² (NIST-certified for FIPS 203/204 compliance).

Internal function tests were conducted by comparing the PQC SDK's outputs with those of PQClean. Each function was tested using randomly generated data to ensure that results matched those of PQClean. Similarly, user-facing functions were tested against `mldsa-native` and `mlkem-native`, which have been NIST-certified for compliance with FIPS 203 and FIPS 204 specifications. When given the same random seed as input, the PQC SDK produces exactly the same results as the native library, ensuring high confidence in its compliance with NIST standards. All tests ran 100,000 times with randomized data to ensure robustness.

2.3 PiQASO QR-TCB Architecture

The architecture of the QR-TCB is based on the core security principle that long-term secret key material must remain exclusively within the TEE boundary, introducing variations for short-term secrets depending on the desired security models. This requires different design choices for KEMs and DSAs.

The integration of PQC within TEEs becomes essential to ensure that secret key material remains protected both cryptographically and at the hardware level. As shown in Figure 2.1, our architecture of a

¹<https://github.com/pqclean/PQClean>

²<https://github.com/pq-code-package>

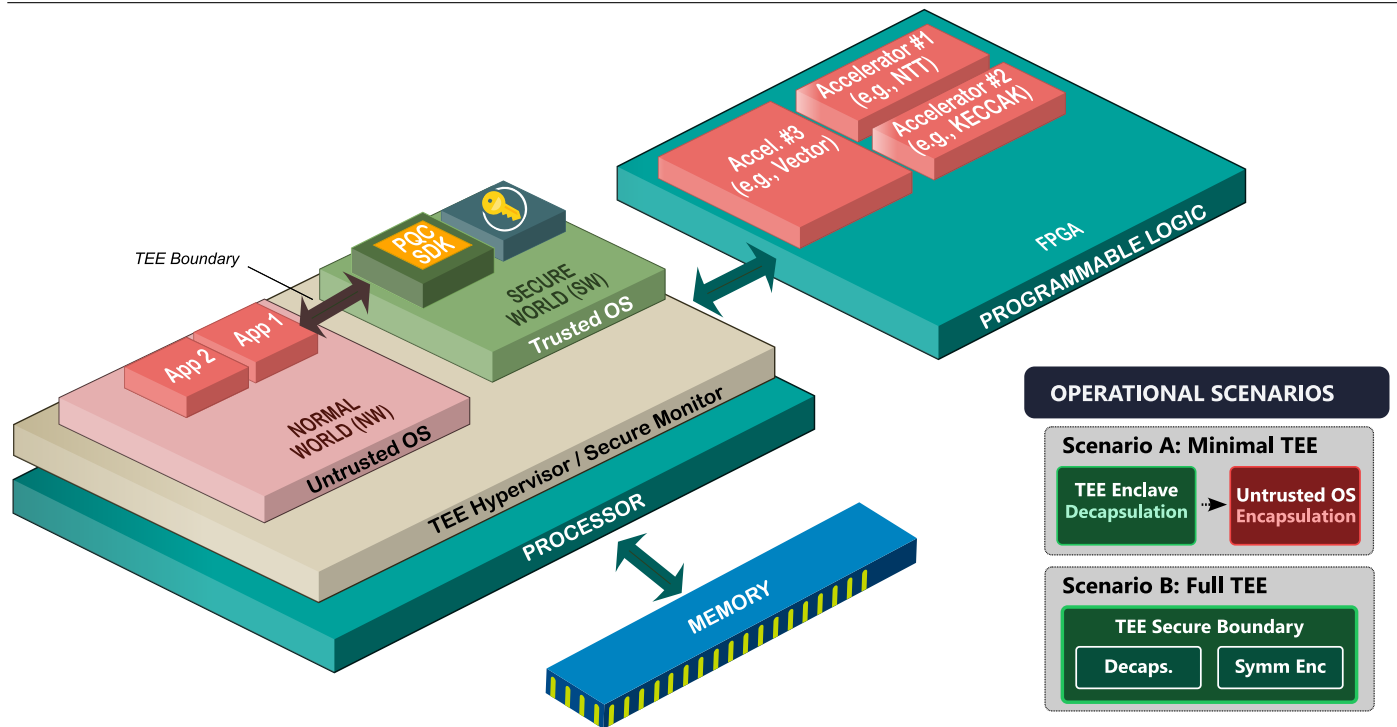


Figure 2.1: QR-TCB Architecture model.

QR-TCB enforces that long-term secrets never leave the TEE boundary, while allowing flexibility for short-term session keys depending on the desired security model, ranging from minimal TEE configurations that reduce resource usage to full TEE deployments where all sensitive operations, including symmetric encryption, occur within the secure enclave. Moreover, since PQC primitives rely heavily on computationally intensive operations such as polynomial arithmetic, the NTT, and KECCAK, hardware acceleration becomes a critical design consideration; accelerators internal to the TEE offer the strongest guarantees, whereas external accelerators improve performance but require careful analysis of data exposure at the TEE boundary.

2.3.1 Key Encapsulation Mechanisms

The KEM secret key should never leave the secure enclave, ensuring that even a compromised operating system or hypervisor cannot extract cryptographic secrets. Key generation and the Decapsulation procedure must execute entirely inside the TEE. This leads to two different scenarios that will be analysed.

Scenario A: Minimal TEE In this configuration, only the Decapsulation operation runs inside the TEE. The Encapsulation procedure, which uses only the public key, executes in the untrusted environment. The resulting session key is handled by the OS, where all data encryption will then occur. This approach minimizes TEE resource usage and latency. The session key, being ephemeral and short-lived, presents a lower risk if briefly exposed during transfer. Moreover, an eventual low-resource client without an available TEE will expose the short-term key to its OS, hence such a session key will not have full TEE security guarantees.

Scenario B: Full TEE Here, both Decapsulation and the subsequent symmetric encryption operate within the TEE. The session key never leaves the secure boundary. This provides stronger guarantees

on the encryption process and ensures complete protection of both key material and plaintext data. Larger data transfer between the OS and TEE is required as every encryption happens inside the TEE.

Hardware Acceleration ML-KEM relies heavily on polynomial arithmetic, the NTT, and the KECCAK function, and hardware accelerators for these operations can significantly improve performance. Accelerators inside the TEE provide the strongest security but require trusted hardware integration. External accelerators offer better performance but require careful analysis of data exposure during calls to the accelerator. The scope of the hardware accelerator will be analysed on a per-operation basis to decide whether it is advantageous to have an accelerator bound to the TEE only, no accelerator in the TEE but at the OS level, or two isolated accelerators for use inside and outside the TEE, in the context of D4.1.

Perfect Forward Secrecy In long-lived deployments, the probability that a secret key will eventually be exposed through implementation flaws, insider threats, or advances in cryptanalysis increases over time. Perfect Forward Secrecy (PFS) addresses this concern by ensuring that the compromise of long-term key material does not retroactively endanger past communications. This is a protocol-specific property that is dependent on which protocol the parties engage in, and ideally, it would be fully executed inside the TEE (as in **Scenario B** above). This property becomes particularly relevant in TEE-based KEM implementations, where the boundary between trusted and untrusted execution creates distinct attack surfaces. Without PFS, a single long-term KEM key-pair is used repeatedly. Compromise of the secret key enables decryption of all past sessions. The TEE must protect one static key-pair indefinitely. With PFS, ephemeral KEM key-pairs are generated per session. Each key generation must occur inside the TEE for each connection. Ephemeral secret keys can be securely deleted after session completion, reducing the window of vulnerability.

2.3.2 Digital Signature Algorithms

For a DSA, the signing key must remain exclusively within the TEE boundary. Unlike KEM, the operational split is straightforward, as the Sign operation requires the secret key and must execute inside the TEE, while the Verify operation uses only the public key and can run anywhere.

Scenario A: Minimal TEE Only the Sign operation executes inside the TEE. Message hashing can occur outside the TEE, with only the hash passed in for signing. This reduces TEE workload but requires trust that the correct message hash is provided. Verification happens entirely in the untrusted environment.

Scenario B: Full TEE Both hashing and signing occur within the TEE. The full message enters the secure enclave, ensuring the signer has complete control over what is being signed. This prevents attacks where a malicious OS substitutes a different message hash.

Hardware Acceleration ML-DSA shares computational characteristics with ML-KEM, relying on polynomial operations, the NTT, and the KECCAK function. The same acceleration trade-offs apply: internal TEE accelerators provide stronger security, while external accelerators offer performance benefits with increased exposure risk. For signatures, the signing path is more critical than verification.

Key Rotation Key compromise allows an attacker to forge future signatures but does not invalidate past signatures, meaning non-repudiation is lost. Key rotation policies can limit the damage window from compromise, and ephemeral signing keys for specific sessions or documents can provide isolation, though this requires per-use key generation inside the TEE. The capability for periodic rotation of the keys based on security policies should be possible.

2.4 PiQASO Hardware Acceleration for PQC

The transition toward Post-Quantum Cryptography (PQC) represents a critical evolution in securing digital infrastructures against future quantum threats, yet it introduces unprecedented computational challenges for embedded systems. Standardized NIST algorithms, such as ML-KEM, ML-DSA, and SLH-DSA, alongside promising code-based candidates like HQC, rely on intensive subroutines that create significant performance bottlenecks in purely software-based implementations. In this regard, the core focus of PiQASO is to *optimize and accelerate the main bottlenecks in the numerical computations required in the context of PQC*. As a primary concern, PiQASO focuses on accelerating the core functionalities of different PQC families, irrespective of the available hardware resources, in order to find the upper bound of acceleration. Specifically, focus is placed on accelerating the **Number Theoretic Transform (NTT)**, which is a core operation of lattice-based crypto, and **Karatsuba** (to be examined in detail in D3.2), which is a core operation of HQC. As a second step, PiQASO will investigate crypto agile sharing, which refers to capability to seamlessly swap or upgrade cryptographic algorithms, keys, and protocols without disrupting underlying system operations.

One core concern is that polynomial arithmetic based on NTT for lattice-based schemes, SHA-3 and SHAKE hash functions, and the high-degree polynomial multiplication required by HQC often exceed the processing capabilities of standard embedded cores. Furthermore, the integration of masking countermeasures to defend against side-channel attacks (SCA) exponentially increases execution time and memory footprints, frequently rendering secure software implementations impractical for resource-constrained edge applications. To address these limitations, PiQASO proposes a comprehensive software/hardware co-design strategy, aimed at offloading performance-critical operations to dedicated hardware accelerators. Empirical analysis reveals that cryptographic hash functions (SHA-3 and SHAKE) account for over 50% of the execution time in ML-KEM and ML-DSA, and exceed 90% in SLH-DSA. Meanwhile, polynomial arithmetic constitutes the remaining primary load, consuming at least 40% of execution time in lattice-based schemes and reaching up to 80–90% in HQC implementations on embedded devices. Furthermore, the integration of side-channel countermeasures significantly exacerbates these bottlenecks. Because hash functions are non-linear, masking them—through techniques such as Domain-Oriented Masking (DOM) or Threshold Implementations (TI)—imposes a massive computational overhead. In contrast, polynomial multiplication remains a linear operation, where complexity grows linearly with the number of shares. Consequently, to enable practical PQC in edge environments, the PiQASO project focuses on two primary hardware acceleration pillars: high-performance, masked hash engines and optimized polynomial multipliers. These accelerators will be integrated into the PQC SDK via dedicated software APIs, ensuring transparent access while preserving the PQC SDK's static memory model, deterministic behavior, and strict standard compliance. Through this hardware/software co-design, we address the non-linear complexity of hash-based masking and the heavy computational load of polynomial arithmetic, ensuring that both masked and unmasked variants meet the rigorous performance requirements of modern edge applications.

2.4.1 Acceleration of Hash primitives

The acceleration of cryptographic hash primitives within the PiQASO framework will be centered around a dedicated hardware core designed to support first-order masking countermeasures natively. This accelerator is intended to serve as a high-performance engine for all hash-based operations across the selected post-quantum schemes, including ML-KEM, ML-DSA, HQC, and SLH-DSA. By offloading these primitives, the architecture directly mitigates the severe performance degradation typical of masked and unprotected software implementations, where non-linear functions like SHA-3 and SHAKE can account for over 90% of the total execution time. The core will be based on a Keccak-p[1600] permutation engine that integrates Domain-Oriented Masking (DOM) to ensure security against first-order side-channel attacks. By employing DOM, the architecture maintains a strict separation between secret shares through independent processing domains, supplemented by intermediate register stages that prevent information leakage caused by combinatorial glitches in the hardware logic. To address the complexity of the non-linear χ (chi) transformation, the design utilizes specialized DOM gadgets that process each share independently. A critical feature of this architecture is the integration of a high-bandwidth randomness dispatcher, which provides the fresh masks required by the DOM gadgets in real-time. This ensures that the injection of randomness does not become a bottleneck, allowing the core to maintain high throughput even during masked operations. The internal state will be managed through two parallel register files, enabling the accelerator to perform round computations on both shares simultaneously within a single clock cycle. The hardware accelerator shall support all FIPS 202 variants and allow for dynamic switching between masked and unmasked execution modes. This last feature is extremely important to guarantee the usability of the accelerator in specific points of the masked algorithm to execute operations which do not require protection. This flexible and high-performance design will be integrated into the PQC SDK framework via the system bus, ensuring full compatibility with the library and providing a scalable security solution for resource-constrained edge devices.

2.4.2 Acceleration of Polynomial Multiplication

The second macro-area of hardware acceleration within PiQASO focuses on polynomial arithmetic, which constitutes the primary computational burden for lattice-based and code-based schemes. For ML-KEM, PiQASO intends to design a specialized hardware accelerator capable of performing the Number Theoretic Transform (NTT), Inverse NTT (INTT), Point-Wise Multiplication (PWM), and polynomial addition and subtraction. To achieve a high degree of efficiency while remaining within the area constraints of embedded platforms, the architecture will feature a parallelized datapath equipped with multiple butterfly units—specifically targeting a four-unit configuration—supported by dedicated high-bandwidth memory banks for polynomial storage. This design ensures that the most time-consuming operations of lattice-based schemes, which typically account for at least 40% of the execution time, are executed with minimal latency and deterministic memory access. A similar architectural approach will be proposed for ML-DSA, featuring a dedicated polynomial engine tailored to its specific algebraic requirements and transform parameters. This accelerator will be optimized to handle the distinct polynomial degrees and modulus operations defined by the ML-DSA standard, ensuring efficient execution of the signature generation and verification processes which are otherwise bottlenecked by software-based arithmetic. For the HQC-KEM scheme, PiQASO will evaluate two distinct architectural strategies to address the high complexity of large-degree polynomial multiplication, which can reach up to 90% of the total execution time on embedded devices. The first approach involves the implementation of a classical Karatsuba-based multiplier, optimized for recursive degree reduction and hardware area efficiency. Alternatively, a more advanced approach based on flattening the Karatsuba algorithm and the internal computations will be explored, as this can lead to a higher upper bound on the acceleration achieved.

2.4.3 Integration of the Accelerators into the ARM Platform

The integration strategy for the proposed post-quantum accelerators focuses on a high-performance hardware/software co-design targeting the Xilinx Zynq UltraScale+ MPSoC platform. The reference processing system is centered around the integrated ARM Cortex-A53 core, which will manage the high-level cryptographic logic and the PQC SDK framework. The specialized hardware accelerators for masked SHA-3/SHAKE, NTT-based arithmetic, and polynomial multiplication will be implemented within the Programmable Logic (PL) section of the SoC. This heterogeneous architecture allows for the offloading of computationally intensive and security-critical tasks to dedicated hardware while maintaining the flexibility of a software-based management layer.

Communication between the ARM processor and the hardware accelerators will be established through a high-performance AXI4 interconnect. To further optimize data transfer efficiency and reduce CPU overhead during the processing of large data structures—such as polynomial matrices in ML-KEM or high-degree polynomials in HQC—the use of a Xilinx Direct Memory Access (DMA) engine may be considered. Implementing a DMA-based approach could enable efficient data transfers between the system memory and the accelerators, potentially maximizing throughput for the most demanding operations.

2.5 “PQC-as-a-Service” for Cloud-based Data Security

The PiQASO Deliverable D3.1 [55, Section 7.5] describes the **PQC-as-a-Service (PQCaaS) framework** as a cloud-based mechanism that enables resource-constrained devices to outsource complex PQ cryptographic operations to a dedicated **PQ Data Encryption Server (PQ-DES)**, thereby allowing such devices to **securely transmit and store data** without performing computationally intensive cryptographic tasks locally. In particular, the PQ-DES serves as the central component of the PQCaaS framework, as it supports all the PQ cryptographic operations for secure management, encryption/decryption (different variants of confidentiality mechanisms), authentication and fine-grained access control, leveraging the PiQASO PQC Ensemble. One of the primary goals of this project is to **implement a PQ-DES based on NIST-standardized PQ algorithms**, ensuring also seamless integration into existing **Common Transmission Systems (CTS)**. Thus, PiQASO aims at designing a PQCaaS solution that provides the **required security guarantees**, while supporting **scalability** and **crypto agility** for quantum-resistant, cloud-based data sharing and storage services, with **minimal impact on end users and service providers**, thereby **enabling a smooth and practical transition towards secure PQ infrastructures and long-term data protection**.

As mentioned in Deliverable D3.1 [55], one of the key challenges in the design of the PiQASO PQCaaS solution is the **secure management and protection of cryptographic keys**, especially in environments requiring fine-grained access control. Since a potential key exposure compromises all associated data stored in the cloud, advanced security properties such as **forward secrecy** and **post-compromise security** become of paramount importance. Therefore, a central innovation of the PiQASO PQCaaS framework is the integration of an **Updatable Public Key Encryption (UPKE)** scheme, which enables **secure key update functionalities** to protect data at rest, while ensuring that forward secrecy is guaranteed. **PiQASO further extends the design space of UPKE constructions by exploring PQ implementations stemming from lattice assumptions**, to support two main modes of operation within the cloud-based PQCaaS model (see also Deliverable D3.1 [55, Section 7.5]), which are based on different trust assumptions and rely on different time-evolving key chains, updated at regular time epochs:

User-specific mode. A forward-secure, end-to-end encrypted data sharing system that uses multiple unlinkable public keys per user and per time epoch, generated via the **Asynchronous Remote Key Mechanism (ARKG)**. This mode of operation considers a specific data sender and multiple

data receivers. In this case, a PQCaaS entity, called the **Key Provider (KP)**, is responsible for authenticating the communicating parties, while the encryption/decryption operations of transferred data are executed by the end users themselves, leveraging the PiQASO PQC Ensemble. Consequently, this is a **fully decentralized mode of operation**, as the cryptographic operations for confidentiality are executed by the endpoints and **no sensitive information is exposed to, or processed by the PQCaaS infrastructure**.

Attribute/policy-based mode. A forward-secure **Attribute-Based Encryption (ABE)** scheme, where time-evolving public keys are linked with access policies, ensuring that only authorized users can decrypt data. This mode also considers a specific data sender and multiple data receivers, where the data sender encrypts data payloads using an attribute-based UPKE scheme. It assumes that the data sender obtains the currently valid ML-KEM public key(s) from the PQCaaS KP, while the eligible receiver obtains the corresponding decryption keys if and only if they meet the eligibility criteria specified by the policy. In this mode of operation, **the PQ-DES is assumed to be a trusted party** responsible for executing the encryption and decryption operations. Consequently, this is a **centralized mode of operation**, as it assumes that the entire **CTS is trusted and it is assigned with the execution of cryptographic operations, as well as with the handling of sensitive data and key material** on behalf of end users.

A high-level description of the PQCaaS component and its core cloud-based functionality has been described in Deliverable D3.1 [55, Figure 7.3]. To realize the PiQASO PQCaaS data management service, **we follow an incremental development approach**. The initial phase introduces a **simplified system model** that captures the core entities and operations, allowing for an easier formal security analysis. At this first stage, the focus is on two primary functionalities of the cloud-based system, namely, **data upload** and **data retrieval**. All stored data is encrypted using a policy-based UPKE scheme, derived from the ML-KEM, ensuring forward secrecy through periodic key updates.

More concretely, in this initial model, we consider a **baseline UPKE construction** as a starting point, in which **multiple ML-KEM public keys are generated from distinct secret keys** by repeatedly executing the ML-KEM key generation function. This naive UPKE construction is the predecessor of more advanced schemes, where multiple public keys are derived from a single secret key, and simplifies the security modelling. We also note that the current security model abstracts away from additional cryptographic operations, such as user authentication, which will be incorporated in later phases where the system will be further refined.

Note that, regarding cloud-based security, the implementation plan of PiQASO consists of two core steps: (i) In the first version of the PQCaaS enabler, the initial version of UPKE is provided. This is part of the **Updatable Hybrid Encryption** approach of PiQASO, under a strong trust assumption model, where *the PQ-DES is considered trusted, and generates and manages the different key pairs between sender and receiver*. This scheme is presented in detail in D3.2, and provides the basis for all corresponding sequence diagrams presented in the context of the use cases. (ii) In the second version, to be detailed in D3.3, we provide the *pure UPKE variant leveraging lattice-based constructions, which weakens the aforementioned trust assumptions, by enabling the keys to be managed by their intended holders*.

2.5.1 Initial Approach to Cloud Storage for Encrypted Data

Figure 2.2 shows a high-level description of the simplified cloud-based data storage solution, concentrating on the **user registration**, **data upload** and **data retrieval** functionalities of the PiQASO PQCaaS service. It assumes a CTS composed of the three entities defined in Deliverable [55, Section 7.5], namely the KP, the PQ-DES and the **Identity Provider (IDP)**, and captures the communication between the three

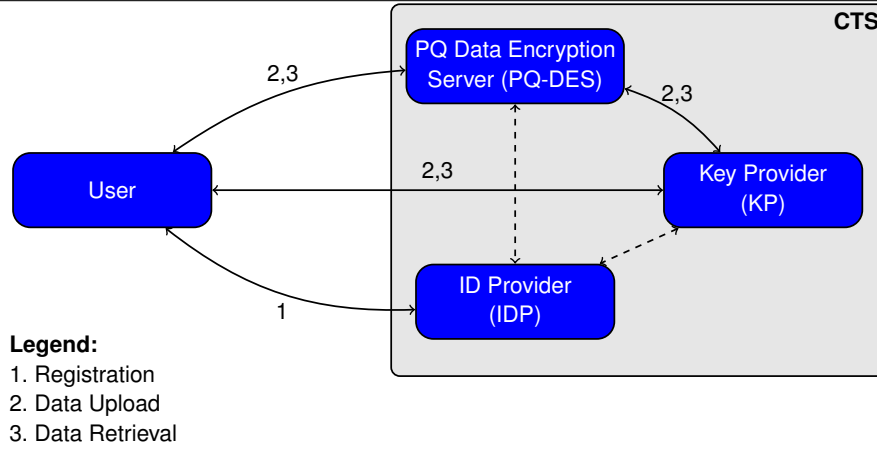


Figure 2.2: System model representation with registration, data upload and data retrieval functionalities

entities, as well as the interaction of the CTS with the end users. Following the earlier description of the two modes of operation, the scheme described in Figure 2.2 assumes that **the IDP, PQ-DES and KP are trusted parties** to facilitate the modelling of the **attribute/policy-based mode of operation**. This initial approach will serve as the baseline for the development of the **complete end-to-end privacy-preserving cloud-based data management service in the next phases of the PiQASO project**, based on an advanced lattice-based UPKE protocol. In the remaining parts of this section, we define the syntax and security properties of the upload and retrieval procedures, and provide a sketch for proving their security. More concretely, our focus is on **formalizing the notion of forward secrecy**, assuming a standard UPKE scheme, as well as the **integration of policy-based access control** mechanisms for allowing access to encrypted data. For simplicity, we further assume that a secure authentication mechanism is in place, that allows the mutual authentication of communicating parties.

Notation. Let v denote a vector of dimension n . We write $v[1, \dots, n] \leftarrow s$ to denote that every entry of the vector v is initialized to the value $s \in \mathcal{S}$. Furthermore, we write $v \leftarrow \text{append}(s)$, to denote that the value s is appended to the vector v , thereby resulting in a vector of dimension $n + 1$, where the last element is $v[n + 1] = s$. Given a database DB_K containing n (ML-KEM) key pairs, we write $\text{DB}_K.\text{sk}$ to denote all secret key entries in the database, i.e., $\{\text{DB}_K[i].\text{sk}\}_{i=1}^n$. We define $\text{DB}_K.\text{pk}$ in the same way.

2.5.1.1 Hybrid Encryption Scheme

Our initial, simplified cloud-based storage system assumes a **Hybrid Encryption** scheme that combines **PQ public key encryption**, with the NIST standardized ML-KEM and **symmetric encryption**, based on the AES-256 block cipher. In particular, the hybrid encryption scheme executes the following two encryption operations: 1. symmetric encryption to encrypt a message m using a symmetric secret key k , and 2. a public key encryption, for encrypting the symmetric key k with an ML-KEM public key pk (key wrapping). The resulting ciphertext c of the hybrid encryption scheme is composed of two parts; a ciphertext c_1 corresponding to the symmetric encryption of the message and a ciphertext c_2 corresponding to the ML-KEM encryption of the symmetric key k . Consequently, the hybrid decryption process requires first an ML-KEM decryption with the corresponding ML-KEM secret key sk to retrieve the symmetric key k , and a symmetric decryption with k to obtain the initial message m . The formal definition of our hybrid encryption notion is described as follows:

Definition 1 (Hybrid Encryption Scheme). *A hybrid encryption scheme is defined as the tuple*

$$\text{HE} = (\text{HE.KeyGen}, \text{HE.Enc}, \text{HE.Dec})$$

where each function is defined in the following way:

- **Key generation** ($\text{HE.KeyGen}(1^\lambda)$). A probabilistic algorithm which, on input the security parameter λ , outputs a secret/public key pair (sk, pk) .
- **Hybrid encryption** ($\text{HE.Enc}(\text{pk}, m)$). A probabilistic encryption algorithm which, on input a public key pk and a message $m \in \{0, 1\}^*$, outputs a ciphertext $c = (c_1, c_2)$.
- **Hybrid decryption** ($\text{HE.Dec}(\text{sk}, c)$). A deterministic decryption algorithm which, on input a secret key sk and a ciphertext $c = (c_1, c_2)$, outputs the message m or the error symbol $\perp$.

Furthermore, we say that the hybrid encryption scheme is δ -correct, if for all messages $m \in \{0, 1\}^*$ and all key pairs (sk, pk) obtained by executing the $\text{HE.KeyGen}()$ function, the following holds:

$$\Pr [\text{HE.Dec}(\text{sk}, \text{HE.Enc}(\text{pk}, m)) = m] \geq 1 - \delta.$$

Since the function $\text{HE.Enc}()$ executes two encryption subroutines, as mentioned above, the final ciphertext c contains two parts c_1, c_2 , denoted by:

$$c_1 = \text{Enc}_{\text{KEM}}(\text{pk}, k) \quad \text{and} \quad c_2 = \text{Enc}_{\text{AES}}(k, m)$$

where the first component is the KEM-based encryption of the symmetric key k and the second component is the symmetric encryption of the actual message m via AES-256.

The following definition captures the IND-CCA2 security notion of the hybrid encryption scheme HE:

Definition 2 (IND-CCA2 security). We say that a hybrid encryption scheme HE of Definition 1 is IND-CCA2 secure, if for any polynomial-time adversary $\mathcal{A}$ it holds:

$$\Pr [\text{Exp}_{\text{HE}}^{\text{ind-cca2}}(1^\lambda) = 1] - \frac{1}{2} \leq \text{negl}_{\text{cca}}(\lambda)$$

where negl_{cca} is a negligible function in the security parameter λ .

The experiment $\text{Exp}_{\mathcal{A}, \text{HE}}^{\text{ind-cca2}}(1^\lambda)$ models the indistinguishability under adaptive chosen-ciphertext attack for the hybrid encryption scheme HE of Definition 1. Formally, this experiment is described as follows:

$\text{Exp}_{\mathcal{A}, \text{HE}}^{\text{ind-cca2}}(1^\lambda) :$

```

1 :  $(\text{sk}, \text{pk}) \leftarrow \text{HE.KeyGen}(1^\lambda)$ 
2 :  $(m_0, m_1, \cdot) \leftarrow \mathcal{A}^{\text{Dec}(\text{sk}, \cdot)}(\text{pk})$ 
3 : if  $|m_0| \neq |m_1|$  then return  $\perp$ 
4 :  $b \xleftarrow{\$} \{0, 1\}$ 
5 :  $c^* \leftarrow \text{HE.Enc}(\text{pk}, m_b)$ 
6 :  $b' \leftarrow \mathcal{A}^{\text{Dec}(\text{sk}, \cdot)}(c^*)$ 
7 : return  $b \stackrel{\$}{=} b'$ 

```

In this experiment, the challenger first generates a key pair (sk, pk) using the HE.KeyGen function. The adversary $\mathcal{A}$ is given access to a decryption oracle $\text{Dec}(\text{sk}, \cdot)$ and outputs two challenge messages (m_0, m_1) of equal size, along with some state information st . Then the challenger samples a random challenge bit $b \in \{0, 1\}$ and computes the challenge ciphertext $c^* \leftarrow \text{HE.Enc}(\text{pk}, m_b)$, which is returned to the adversary. After receiving the challenge ciphertext c^* , the adversary may continue to query the decryption oracle, with the restriction that it cannot request the decryption of the challenge ciphertext c^* itself. Finally, the adversary outputs a guess b' . The experiment returns 1 if $b' = b$, and 0 otherwise.

2.5.1.2 A Forward-Secure Cloud Storage Scheme

We present a high-level description of our initial, simplified cloud storage scheme. In this preliminary construction, **our main objective is to model the forward secrecy property**, considering a **naive UPKE scheme**, where ML-KEM key pairs are periodically updated by re-executing the key generation function of ML-KEM. As a result, the compromise of a secret key at a specific time does not compromise the confidentiality of data that was encrypted and stored using keys from previous epochs. To simplify our modelling analysis, we further assume that the cloud service performs **key update at regular, but fixed time intervals**. In addition, **our model includes policy-based access control**, allowing end users to access encrypted data only when a specified policy is satisfied. Consequently, each ciphertext is associated with an attribute-based policy enforced by the cloud service.

Remark. *In our initial cloud-based storage scheme we consider a HE scheme based on a naive approach for modelling the UPKE functionality, where ML-KEM keys are replaced by new ones at each epoch. In later phases of the project, we will replace this ML-KEM-based approach with a novel, lattice-based UPKE, where multiple public keys are derived from a single secret key.*

Syntax of Cloud Storage Scheme. We define a cloud-based storage scheme as a set of four probabilistic polynomial time algorithms (Setup, Upload, Update, Retrieve), together with a hybrid encryption scheme as in Definition 1, namely $HE = (HE.KeyGen, HE.Enc, HE.Dec)$. We give a brief description of each function and the involved entities below:

System Setup ($Setup(1^\lambda, n)$). On input the security parameter λ and the number of users n , this procedure returns four initialized databases (lists); a database for files DB_f , a database for keys DB_K , a database for policies DB_P and a database for epochs DB_e . The databases DB_f and DB_P are initialized as empty lists. The database DB_K contains n pairs of ML-KEM encryption keys for the HE scheme, as well as the epoch related to each key pair. The database DB_e stores all the epochs for every key pair. We note that the databases DB_P , DB_f , DB_e and the public keys contained in DB_K (i.e., $DB_K.pk$) are publicly available after generation.

Data Upload ($Upload(f, P)$). On input a file $f \in F$, representing the user's data payload, and a policy P , the procedure possibly modifies the state of the databases DB_f and DB_P and outputs a file identification string ID_f , or the symbol $\perp$. This procedure is executed by the PQ-DES, the entity responsible for performing the cryptographic operations, such as the encryption of data on behalf of users, using the $HE.Enc$ function of Definition 1.

Key Update ($Update()$). This procedure updates ML-KEM public key pairs in the database DB_K , as the epoch evolves. Since we are assuming a naive UPKE approach, an update of n key pairs implies the re-execution of the KeyGen function of the HE scheme n times, and the replacement of the old key pairs with the new ones. To further simplify notation, we will use $DB_K.pk^e$ to indicate all public keys of time epoch e . In particular, we have:

$$DB_K.pk^e = \{DB_K[i][e].pk\}_{i=1}^n$$

meaning all public keys of the n users, related to the epoch e . In addition, we recall the notation $DB_K.pk$, indicating all public keys of all epochs. The Update procedure is executed by the KP, upon request from the PQ-DES.

Data Retrieval ($Retrieve(ID_U, ID_f)$). On input a user identification string ID_U and a file identification string ID_f , the procedure outputs either the file f or the error symbol $\perp$. This function is executed by the PQ-DES, which is responsible for decrypting stored files using the $HE.Dec$ function of Definition 1.

When a file f with identification string ID_f is encrypted at the time epoch e , we write $DB_e[ID_f] = e$. Furthermore, since ML-KEM key pairs are stored in DB_K together with the associated time epoch, the database DB_K corresponds to a table. More concretely, for the ML-KEM key pair of a user with identification string ID_U and epoch e we denote the corresponding key pair by $(sk_{ID_U}^e, pk_{ID_U}^e)$ and this key pair is stored in entry $[ID_U][e]$, i.e., $DB_K[ID_U][e] = (sk_{ID_U}^e, pk_{ID_U}^e)$. The current epoch is indicated by a global value, which is incremented every time the epoch evolves and it is included in the policy associated with the user with identity ID_U . Based on this syntax, we obtain the following definition of correctness for our simplified cloud-based storage scheme.

Definition 3 (Correctness). *From the above algorithms that compose the cloud-based storage scheme, the following notion of correctness is satisfied:*

$$\Pr \left[\text{Retrieve}(ID_U, \text{Upload}(f, P)) = f \mid ID_U \in P \right] \geq 1 - \delta,$$

for $\{DB_f, DB_P, DB_K, DB_e\} \leftarrow \text{Setup}(1^\lambda, n)$, any file $f \in F$ and any policy P , where the Upload is executed at any time epoch e , and for a negligible function $\delta(\lambda)$.

Instantiation based on Hybrid Encryption. We proceed with a detailed description of the instantiation of our simplified cloud-based storage scheme. Let $HE = (HE.\text{KeyGen}, HE.\text{Enc}, HE.\text{Dec})$ be an IND-CCA2 hybrid encryption scheme as in Definition 1. We assume that a user has obtained an identification string ID_U from the IDP during the registration phase, and it has already authenticated with the CTS and particularly with the IDP, hence a secure communication channel has already been established between the user and the CTS. Recall that this authentication phase is excluded from our initial model.

A user with identity ID_U who wishes to store their encrypted data in the cloud proceeds as follows. It first generates a random (symmetric) secret key k , by applying a Key Derivation Function (KDF) provided by the PiQASO PQC Ensemble, which is used to encrypt their file f (data payload) with symmetric encryption based on the AES-256 blockcipher. The user encrypts the symmetric key k with the ML-KEM public key pk_{CTS} of the CTS, using the $HE.\text{Enc}$ function of the hybrid encryption scheme. Thus, a ciphertext $c = (c_1, c_2)$ is produced, such that:

$$c = (c_1, c_2) = (\text{Enc}_{KEM}(pk_{CTS}, k), \text{Enc}_{AES}(k, f))$$

The ciphertext c is forwarded to the PQ-DES. We note that in order to achieve the intended support for legacy protocols, in a scenario where a user is unable to execute ML-KEM encryption to produce c_1 , the secret key k can be encrypted using any other KEM mechanism supported by both the user and the CTS, or send the key to the PQ-DES over an existing secure communication channel (e.g., TLS).

After receiving the ciphertext c , the PQ-DES communicates with the KP, in order for the KP to generate an ML-KEM key pair $(sk_{ID_U}^e, pk_{ID_U}^e)$ for the user with identity ID_U , for the current epoch e , using the PiQASO PQC Ensemble. The PQ-DES obtains the secret key k generated by the user, by decrypting the ciphertext c_1 using its ML-KEM secret key sk_{CTS} , or by receiving it over the existing secure communication channel. Then, it re-encrypts k using the new ML-KEM public key $pk_{ID_U}^e$ on behalf of the user ID_U , producing a ciphertext $c'_1 = \text{Enc}_{KEM}(pk_{ID_U}^e, k)$. The PQ-DES stores the new ML-KEM key pair in the database entry $DB_K[ID_U][e]$ and the ciphertexts (c'_1, c_2) in the database entry $DB_f[ID_U]$. Hence, in order for user ID_U to gain access to their encrypted data f , they first need to obtain access to the symmetric key k . In addition, in the case where the ML-KEM key pair $(sk_{ID_U}^e, pk_{ID_U}^e)$ is updated, the user should still be able to access their encrypted data. This implies that the epoch e should be part of the policy associated with the user ID_U , allowing the user to execute the ML-KEM key generation function to retrieve ML-KEM key pairs at previous epochs.

We proceed with the detailed description of the algorithms (Setup, Upload, Update, Retrieve). The Setup algorithm initializes a global counter $count_f$, to generate unique identifiers for uploaded files. This counter

is initially set to 0 and incremented with each new upload, however, any method that ensures uniqueness of file identifiers can be used instead. In addition, three empty databases are created: DB_f for storing encrypted data, DB_P for access policies and DB_e for storing the epochs. An additional database DB_K is also created and initialized with one ML-KEM key pair per user, at epoch 0. Each entry $DB_K[i][0]$ is generated by executing the ML-KEM key generation algorithm. The algorithm outputs the four databases and recall that all databases are public, except for the secret key entries in $DB_K.sk$. The Setup algorithm is formally described as follows:

$Setup(1^\lambda, n) :$

```

1 : global  $count_f \leftarrow 0, t \leftarrow 0, DB_f, DB_P, DB_e, DB_K[1, \dots, n][0] \leftarrow (\epsilon, \epsilon)$ 
2 : for  $i = 1$  to  $n$  do
3 :    $DB_K[i][0] \leftarrow HE.KeyGen(1^\lambda)$ 
4 : return  $(DB_f, DB_P, DB_e, DB_K)$ 

```

The Upload algorithm starts by verifying that the access policy P is non-empty. In the case where P is empty, the algorithm returns the error symbol $\perp$. This ensures that every uploaded file can be accessed by at least the user who submitted the file to the CTS. If the access policy P is valid, a new file identifier ID_f is generated by incrementing the global counter $count_f$. Then for each user $j \in P$, meaning for every user included in the access policy P , the function $HE.Enc$ of the hybrid encryption scheme is executed by the PQ-DES, using the public key associated with the user j and the file f . The corresponding public key is retrieved from the j^{th} entry of the database DB_K . Since the file f is encrypted with the symmetric key k_j , generated by the user j , the PQ-DES encrypts this symmetric key with the user's public key for the current epoch e . The resulting ciphertext, together with the associated user identifier, are stored in the file database DB_f with index ID_f . More specifically, the pair (j, c) , consisting of the user identifier and the ciphertext c , is appended to the database entry $DB_f[ID_f]$. After all encryptions are performed, the access policy P is recorded in the database DB_P and the corresponding epoch e in the database DB_e , under the same index ID_f . Thus, for a file with identifier ID_f and policy P , the entry $DB_f[ID_f]$ contains all encryptions of f , $DB_P[ID_f] = P$ stores the associated access policy, and $DB_e[ID_f] = e$ corresponds to the epoch associated with the file f . Finally, the algorithm outputs the file identifier ID_f . The Upload procedure is summarized in the following steps:

$Upload(f, P) :$

```

1 : if  $P = \emptyset$  then return  $\perp$ 
2 :  $ID_f \leftarrow count_f + 1$ 
3 : for all  $j \in P$  do
4 :    $C \leftarrow HE.Enc(DB_K[j].pk^t, f)$ 
5 :    $DB_f[ID_f] \leftarrow append((j, C))$ 
6 :  $DB_P[ID_f] \leftarrow P$ 
7 :  $DB_e[ID_f] \leftarrow t$ 
8 : return  $ID_f$ 

```

The Retrieve algorithm first checks whether the request is valid by verifying that the file identifier ID_f corresponds to an existing entry in the file database DB_f and that the user identifier ID_U is valid. If one of these checks fails, the algorithm returns the error symbol $\perp$. Otherwise, the algorithm retrieves the access policy associated with the requested file from the policy database DB_P . Access control is then enforced by checking whether the user identifier ID_U belongs to the policy P , since only authorized users are allowed to access the file. In this case, the ciphertext associated with user ID_U is decrypted using

HE.Dec function of Definition 1 and the secret key $DB_K[ID_U].sk$, and the original file in plaintext is returned. Otherwise, the algorithm outputs an error. The Retrieve procedure is summarized as follows:

Retrieve(ID_U, ID_f) :

```

1: if  $ID_f > |DB_f|$  or  $ID_U > n$  then return  $\perp$  //here, equivalently, can be checked if  $ID_f$  is a valid ID string
2:  $P \leftarrow DB_P[ID_f]$ 
3:  $e \leftarrow DB_e[ID_f]$ 
4: if  $ID_U \notin P$  then return  $\perp$ 
5: for  $tpl \in DB_f[ID_f]$  do
6:   if  $tpl_1 == ID_U$  then
7:      $f \leftarrow HE.Dec(DB_K[ID_U].sk^e, tpl_2)$ 
8:     return  $f$ 
9: return  $\perp$ 

```

The Update algorithm follows the naive approach of UPKE schemes, where updating the ML-KEM key pairs requires the KP to re-execute the KeyGen function of the hybrid encryption scheme n times, assuming there are n users in the system. In this case, a new ML-KEM key pair (sk_i^e, pk_i^e) is generated for each user at the new epoch e and stored in the database DB_K , specifically in the entry $DB_K[i][e]$. The procedure is described in detail as follows:

Update() :

```

1:  $t++$ 
2: for  $i = 1$  to  $n$  do
3:    $sk_i^e, pk_i^e \leftarrow HE.KeyGen(1^\lambda)$ 
4:    $DB_K[i][e] \leftarrow append((sk_i^e, pk_i^e))$ 

```

We note that the correctness of the proposed instantiation follows directly from the correctness of the underlying hybrid encryption scheme described in Definition 1.

2.5.1.3 Definition of Forward Secrecy for the Cloud Storage Scheme

We now introduce an initial security model for the simplified cloud-based storage scheme, designed to capture the notion of forward secrecy, based on the hybrid encryption scheme of Definition 1. The corresponding experiment, denoted by $Exp_{\mathcal{A}}^{\text{forw-sec}}(1^\lambda, n)$ is intended to formally model and evaluate this security property.

During the setup phase, the databases DB_f , DB_K , DB_e and DB_P , each of size n , are initialized. Additionally, a vector $corr$ is created, indicating the corrupted users, and the current epoch e is initialized to 0. The adversary submits two candidate files along with a challenge policy P^* . One of the two files is selected, uploaded and stored in the cloud system, while the corresponding file identifier ID_f^* is returned to the adversary. The adversary continues to have access to the stored files and may adaptively perform additional upload and retrieval queries, thereby modelling adaptive security. Throughout the experiment, the adversary is also allowed to interact with the following oracles:

Compromise oracle ($\mathcal{O}_{\text{comp}}$). On input an index i identifying a user, it returns the secret key sk_i and marks the user as corrupted in the corruption vector $corr$.

Upload oracle ($\mathcal{O}_{\text{Upload}}$). On input a policy and a file, it executes the Upload algorithm of the cloud-based storage scheme.

Retrieve oracle ($\mathcal{O}_{\text{Retrieve}}$). On input a file identity string and a user identity string, it executes the Retrieve algorithm of the cloud-based storage scheme, under the condition that, when the challenge identity string is provided, the user identity is not part of the policy related to the challenge identity. This is formally written as $\text{Retrieve}(i, \text{ID}_f^*) = \perp$, for each $i \in P^*$.

Update oracle ($\mathcal{O}_{\text{Update}}$). The adversary is allowed to observe encryptions and obtain decryptions across different epochs. Further, the adversary can query the update oracle $\mathcal{O}_{\text{Update}}$, which updates all keys for the current epoch and increments the epoch counter e . To capture the notion of forward secrecy, the adversary is given additional access to the secret keys corresponding to the current epoch at the end of the experiment, even though these keys belong to a different epoch from the one used to generate the challenge ciphertext. This assumption reflects the core principle of forward secrecy, where compromising current keys at a given epoch should not compromise the confidentiality of data encrypted in previous epochs.

The adversary wins the game if it can successfully distinguish which of the two files was selected for upload, with probability greater than random guessing. To avoid trivial success results of the experiment, the model enforces an additional restriction, requiring that the adversary has not corrupted any user included in the access policy of the challenge file. The $\text{Exp}_{\mathcal{A}}^{\text{forw-sec}}(1^\lambda, n)$ experiment is formally described as follows:

$\text{Exp}_{\mathcal{A}}^{\text{forw-sec}}(1^\lambda, n) :$

```

1 :  $\text{DB}_f, \text{DB}_P, \text{DB}_K, \text{DB}_e \leftarrow \text{Setup}(1^\lambda, n)$ 
2 :  $\text{corr}[1, \dots, n] = \text{false}$ 
3 :  $e = 0$ 
4 :  $(f_0, f_1, P^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Upload}}, \mathcal{O}_{\text{Retrieve}}, \mathcal{O}_{\text{comp}}, \mathcal{O}_{\text{Update}}}(\text{DB}_f, \text{DB}_P, \text{DB}_e, \text{DB}_K.\text{pk})$ 
5 :  $b \xleftarrow{\$} \{0, 1\}$ 
6 :  $st \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Upload}}, \mathcal{O}_{\text{Retrieve}}, \mathcal{O}_{\text{comp}}, \mathcal{O}_{\text{Update}}}(\text{DB}_f, \text{DB}_P, \text{DB}_e, \text{DB}_K.\text{pk})$ 
7 :  $\text{ID}_f^* \leftarrow \text{Upload}(f_b, P^*)$ 
8 :  $\text{Update}(\text{DB}_K.\text{pk}^e)$ 
9 :  $b' \leftarrow \mathcal{A}(\text{ID}_f^*, \text{DB}_f, \text{DB}_P, \text{DB}_e, \text{DB}_K.\text{pk}, \text{DB}_K.\text{sk}^e)$ 
10 : return  $b \stackrel{?}{=} b' \wedge \forall i \in P^*. \neg \text{corr}[i]$ 

```

Definition 4 (Forward-Secure Cloud Storage). *An updatable cloud storage server is said to be forward-secure if, for any probabilistic polynomial-time algorithm $\mathcal{A}$, we have:*

$$\left| \Pr \left[\text{Exp}_{\mathcal{A}}^{\text{forw-sec}}(1^\lambda, n) = 1 \right] - \frac{1}{2} \right| \leq \text{negl}(\lambda)$$

for a fixed number of users n and a negligible function negl in the security parameter λ .

We note that **the modelling approach analysed in this section represents a preliminary stage** and serves as the baseline for the security analysis of the PiQASO cloud-based data management system with forward secrecy, based on a predecessor quantum-secure UPKE scheme. In this initial phase, the design focuses on **establishing the core architecture, identifying the entities involved in the system, and assessing the feasibility of the proposed approach**. In later phases of the PiQASO project, **this**

preliminary framework will be further refined, and an extended security analysis will be provided. This analysis will aim to include additional functionalities, capture a wider range of security properties, and evaluate the system against more sophisticated attack vectors, thereby ensuring that it achieves the envisioned security guarantees.

Chapter 3

Methodology for Elicitation of MVP, Requirements, and KPIs

This Chapter is dedicated to detailing the methodology followed by PiQASO for the definition of the technical requirements of the use case demonstrators, as well as the **Key Performance Indicators (KPIs)** that need to be fulfilled for evaluating performance of the PQC enablers of PiQASO in the context of the use case demonstrators. We first provide details on the methodology for the design of the PiQASO **Minimum Viable Product (MVP)**, which comprises the core functionalities that enable the post-quantum transition of a service graph chain, regardless of application domain. Then, we continue with the elicitation of the technical and use case requirements that support the development of the core MVP features.

Within this framework, the methodology described herein has been applied to design the complete PiQASO landscape in terms of both research and specifications. The relationship between the technical and functional requirements and the various use case needs and scenarios is detailed throughout the Chapters dedicated to all PiQASO use case demonstrators in the remainder of this deliverable. Furthermore, the design of the PiQASO MVP has been carefully planned to incorporate the activities scheduled for the project demonstrators. This systematic approach ensures a coherent and well-structured foundation for the subsequent stages of PiQASO.

3.1 Methodology for MVP Design

As a term introduced by Frank Robinson in 2001, the "Minimum Viable Product" (MVP) pertains to an agile and lean approach for the validated planning and design of a product. An MVP constitutes a product version with an adequate number of features that can satisfy the needs of initial users, while targeting at offering input for further product improvements in the future. By providing early feedback during product development, an MVP has a lower cost compared to the implementation of a wider stack of features and the subsequent collection of feedback at the end of the implementation process of the product. In addition, an MVP can help newly established companies to discover business opportunities by experimenting on the reactions of customers while trying different business models.

The use cases of the PiQASO project will contribute to the design of an effective MVP. Even from the initial phase of WP2, they have given input about their activities that should be described in the PiQASO methodology. The demonstrators' engagement contributed to the establishment of a clear definition of the project's scope and purpose. This was achieved by delving into the actual benefits that the proposed solution would offer to the end users. The consortium engaged in a collaborative process, where the demonstrators shared their insights and the technical, research, and academic partners analysed the information. This process allowed the consortium to answer the fundamental question of "What Problem

You're Solving." Additionally, by describing the "as-is" and "to-be" scenarios for each demonstrator, the consortium identified areas where the project vision could be refined and demonstrated how the existing workflows could be enhanced with the intervention of PiQASO. Moreover, by acknowledging existing gaps, PiQASO also aims to develop additional valuable services that provide competitive advantages to end-user organisations.

3.1.1 Requirements Definition Process

The process behind the definition of the core value propositions of PiQASO was initiated with the beginning of the project with the collaboration of all technical and use case partners. This process, which was discussed in D3.1 [55], entailed the identification of the core enablers for PQC migration (defined in D3.1 and expanded in Chapter 2 of this deliverable), and an analysis of the State-of-the-Art for each of the identified project innovations along with an in-depth analysis of the existing vulnerabilities at PQC level and adversarial models considered in mixed-criticality heterogeneous service graph chains, in alignment with the current PQC standardization landscape. This analysis was particularly important given the research nature of PiQASO, as it allowed the consortium to gain insights into the competition and understand the current state of the market. This information will drive the definition of future pathways for exploiting the project's outputs and defining the go-to-market strategy for both the framework itself and its individual assets.

Moving forward with the MVP definition, the consortium proceeded to identify the stakeholders within the supply chain context. They also determined the specific actors and entities involved in the envisioned use cases, along with their respective goals. This exercise aimed to gain a deeper understanding of the intended beneficiaries for whom the consortium is developing the solution. To achieve this goal, it is necessary to gather and prioritise requirements. Within the context of PiQASO, requirements are gathered in two ways. Firstly, technical requirements are derived from the technical partners within the consortium who are responsible for designing and developing the overall solution (D3.1). Secondly, requirements are defined based on the use cases or user stories provided by the project's demonstrators, which will be detailed throughout the remainder of this deliverable. Once all the requirements have been collected, the partner in charge of requirements elicitation, with the consensus of the entire consortium, will prioritise them.

As the project progresses, an essential component of the MVP definition is the implementation of the "Build, Test, and Learn" cycle. This cycle is integrated into the development of the core technical Work Packages (WP4-WP5), which involve the creation of both early and advanced versions of all project modules, including the PiQASO framework itself. Through an iterative process, all technical partners will develop their respective modules and conduct testing at early and mature trial sites (demonstrators). This approach aims to maximise the value added both to each module and the whole PiQASO framework.

The MVP definition process outlined in this section aims to provide a specific and comprehensive depiction of the PiQASO solution. This includes a detailed examination and analysis of the technical, functional, and security requirements of the various architectural components involved.

At its core, the MVP aligns with the overall vision of the consortium and the adopted product development process, taking into consideration the expectations of the users. Its purpose is to provide tangible value, validate methodological ideas and hypotheses, and serve as a guide for design and development activities throughout the project implementation. The PiQASO MVP is expected to play a crucial role in directing and shaping the ongoing work within the project.

3.2 Requirements Elicitation Methodology

The requirements elicitation methodology adopted in PiQASO follows a top-down approach to ensure that the framework requirements are grounded in operational reality and remain reusable across heterogeneous infrastructures. Each use case demonstrator first defines the core **motivation behind the need for leveraging the capabilities of PiQASO to achieve PQC transition**. This goes beyond the advent of quantum computing that necessitates the transition from traditional crypto to post-quantum cryptographic capabilities, but also considers the specific circumstances of each use case, such as network and environmental conditions, or computational capabilities of the devices and entities comprising the service graph chain. Then, based on these, each use case demonstrator provides a refined version of their **Challenges and Needs from PiQASO**, which constitutes the list of challenges pertaining to post-quantum transition that need to be addressed by PiQASO.

Then, after these needs have been defined, each use case provides a set of **"To-Be" Reference Scenarios**, which constitute operational storylines constructed in order to address these needs. These detail the PiQASO offerings that need to be integrated into the architectural environment of the target use case, as well as their positioning in the overall architecture, in order to fulfil an operation pertaining to post-quantum securing an operation or application of the target use case. Each scenario corresponds to a **User Story**, which refers to the detailed action workflow that needs to be followed for the validation of each scenario. These are accompanied by detailed **sequence diagrams**, which provide a comprehensive overview of these actions, and will guide the evaluation activities that need to be taken in the context of each use case. The success of this evaluation is assessed based on the identified KPIs, which are defined according to the methodology detailed in Section 3.3.

3.2.1 Technical Requirements Specification Process

Next, we provide further details on the process followed for the **definition of requirements in the context of the PiQASO use cases**. A requirement is a statement that translates or expresses a need and its associated constraints and conditions, with the purpose of transforming a stakeholder-oriented view of desired services into a technical view of a product or framework capability that can deliver those services. In PiQASO, this translation is essential for ensuring traceability from operational needs (captured through scenario and storylines) to technical design and evaluation activities.

Requirements elicitation constitutes a foundational activity for developing the business value and technical relevance of the PiQASO framework. Since requirements become the reference point for defining, implementing, and validating framework capabilities, they must be comprehensive, transparent, and internally consistent. Requirements elicitation is therefore treated as a structured and iterative process rather than a one-time event. During early phases, requirements are captured through stakeholder engagement mechanisms such as interviews and requirements workshops. As requirements are subsequently used to define architectures, derive design constraints, and validate prototypes, additional gaps and ambiguities are often revealed, requiring refinement and the elicitation of further detail. The following subsections provide an overview of the concrete methods employed within PiQASO for defining and refining stakeholder requirements, and for ensuring their prioritisation and validation in collaboration with end-users and stakeholders.

The PiQASO project has employed an agile methodology, characterised by its iterative nature. This methodology emphasises clear communication and understanding between the business, technical, and scientific aspects of the project. It establishes transparent expectations at the project's commencement and at each milestone, fostering collaboration and progress. The PiQASO approach to system requirement specification initiates with individual interviews conducted with the consortium technical partners.

These interviews were carried out using online tools, resulting in the collection of raw requirements. Raw requirements refer to requirements that have not undergone analysis or been formally documented in a well-structured requirement notation. The collected raw requirements are then subjected to an iterative internal process led by the PiQASO system architects to yield more refined results.

During this phase, brainstorming techniques are employed through ad-hoc calls to further enhance the requirements. The outcomes of these brainstorming sessions are subsequently compared with the technical aspects of the system and linked to one or more of the specified value propositions of the project. This iterative process ultimately leads to the identification and formulation of the necessary technical requirements.

Nevertheless, it is important to acknowledge that requirements derived from interviews can occasionally be unclear. This ambiguity may arise due to potential misinterpretation of stakeholder needs by the system architects, or when stakeholders struggle to comprehend certain questions or lack technical expertise to provide accurate answers. As a result, alongside the process of requirements collection, the PiQASO academic partners have conducted a comprehensive document analysis. This analysis takes the form of a State-of-the-Art review, including an extensive list of industry practices and relevant literature. The purpose of this analysis is twofold: to validate the specified requirements and to identify applicable standards and constraints. By conducting this analysis, the aim is to enhance the quality of the requirements generated within the project.

3.2.2 Use Case Requirements Specification Process

In addition to the overall input on technical requirements, the use case partners provided a detailed description of their user stories and a technical explanation of how they intended to utilize the PiQASO framework. They also specified the core functionalities they planned to leverage. This allowed for a more precise alignment of requirements with the demonstrators, providing opportunities for further research and enabling the demonstrators to explore multiple implementations, if time and resources permit.

The use case partners collaborated closely with the research partners to refine the requirements and elaborate on the technical details derived from the narratives. This followed an approach commonly observed in agile projects. The refinement process involved translating the narratives into specific user stories. User stories are concise units of ideas used to provide high-level descriptions of requirements. They depict a particular feature from the perspective of an end user. User stories are designed to be easily understandable and expressible by non-technical partners. Despite being short, typically consisting of a single sentence, user stories possess the unique ability to be self-explanatory and contain sufficient information to describe the requirement. This allows developers to provide a reasonable estimate of the effort required for implementation.

For each of the distinct PiQASO Use Cases, the demonstrator partners were initially requested to provide user stories that describe the "to-be" reference scenario. To elicit requirements from these user stories, the perspective that a user story serves as a well-defined requirement is embraced, in the sense that:

- Emphasises the perspective of a specific role that will utilise or be affected by the solution.
- Defines the requirement using language that is meaningful to that role.
- Clarifies the reasoning behind the requirement.
- Facilitates the definition of high-level requirements without delving into detailed specifics very early.
- Takes into account the user's goals and the business value associated with each requirement.

These aspects are captured using a straightforward textual template to create a comprehensive sentence. While various templates are available, the Connextra template is employed by 70% of practitioners [18].

User Story ID: User Story Title

As a <type of user>, I want to <some goal or action> so that <some reason or benefit>

Note that the core motivation behind this user story definition is to *demonstrate the system element and the actions that need to be performed towards achieving a goal, such as the post-quantum protection of a set of data*. This reasoning is followed for the formulation of the user stories in the context of all use case demonstrators, and the goal is for this information to be conveyed, regardless of if this specific template is followed or not.

3.3 Evaluation Plan and Application-Agnostic KPIs

After having defined the requirements corresponding to the PiQASO use cases, here we provide further details on the approach followed by PiQASO for defining the evaluation criteria based on which the fulfilment of these requirements will be evaluated. Specifically, the evaluation plan of PiQASO is twofold: (i) Each individual PiQASO component and PQC scheme will be benchmarked based on a set of **test cases**, without being placed in the context of specific application domains. (ii) The PiQASO Framework will be evaluated in the context of **all PiQASO use case demonstrators** based on the set of application scenarios and user stories defined throughout the remainder of this deliverable. In this regard, this Chapter is dedicated to the definition of the **evaluation metrics** based on which the performance of the PiQASO offerings will be assessed. These include both **Common Evaluation Metrics/KPIs**, which apply to all use cases and are applied based on different acceptance criteria, and **Application-Specific Evaluation Metrics/KPIs**, which refer to the set of metrics that are specific to each use case demonstrator.

Note that, regarding both the common KPIs, as well as the application-specific KPIs detailed in each dedicated use case Chapter, *these are initial validation targets stemming from estimations based on the operational environment of the use case demonstrators, so as to not affect the security profile of the application. These may be refined after baseline values are collected, i.e., results pertaining to operations prior to the integration and application of PiQASO functionalities*. In addition, to the extent possible, *the acceptance criteria detailed in the context of the use cases are fully aligned not only with industrial requirements, but also specifications defined in the standards, where applicable*. For instance, in the Automotive use case, the acceptance criteria are driven by the **ETSI** requirements and the **CAR 2 CAR Communication Consortium**, where it is defined that a safety critical application must be concluded within 250ms. Similar approaches have been followed for the rest of the use cases, and particularly those representing safety-critical domains, such as the Aerospace domain.

Also, in the following tables, note that the computational overhead induced by running trusted applications in the QR-TCB is expressed in terms of a percentage. *This refers to the percentage against the baseline of running the same PQ operations outside of the Trusted Execution Environment (TEE)*.

Table 3.1: Common evaluation metrics for all PiQASO use cases

Evaluation Metric	Description
Latency	Considering the need of heterogeneous mixed-criticality applications for time-sensitive execution of commands, as well as the limited computational capability of endpoint devices, the latency of PQC operations should be evaluated, whether the operations are executed on the devices through the PQC SDK or are offloaded through the PQCaaS.

Bandwidth	In mixed-criticality services and applications, it is important to consider the capacity of the communication channel between endpoint devices and the backend infrastructure. This should be evaluated against the size of transferred information, including the size of the signatures introduced by the employed PQ signature schemes.
E2E crypto execution	This metric entails the measurement of the time required for the entire end-to-end execution of a PQ crypto process. This begins after the collection of the data by the edge device and may include the encryption process, transmission to the backend, storage on the QR-Ledger, and decryption processes.
PQC computation overhead (cycles or time)	As the impact of the introduction of PQC on the normal operations of a device needs to be evaluated, this metric entails the measurement of the computational overhead, either in terms of time or computational cycles.
Network Overhead	The types of mixed-criticality heterogeneous application domains examined in PiQASO leverage communication networks (e.g., Wi-Fi, 4G/5G) for carrying operational and telemetry data. These may be subject to environmental conditions, delays, packet losses, fragmentation, etc. This metric entails measuring the network overhead induced when transmitting PQC encrypted data, considering all aforementioned factors.
Number of mitigation strategies (PQ-TLS)	Number of mitigation strategies to be tested in case the PQ-TLS protocol is used for establishing secure and authenticated channels, and depending on whether the PQC SDK is integrated at the edge or the protocols go through the PQCaaS.
Number of mitigation strategies (IKE)	Number of mitigation strategies to be tested in the context of the IKE protocol, which is the foundational mechanism used in IPsec to set up security associations.
Hardware-based Acceleration	Reduction in the execution time of PQC primitives and cryptographic operations thanks to the introduction of the hardware-based acceleration and algorithmic optimization capabilities of PiQASO, compared against baseline implementations of these cryptographic operations.

Table 3.2: Acceptance criteria for the common evaluation metrics in the Automotive use case (UC1)

Evaluation Metric	Acceptance Criteria
Latency	Single sign ≤ 50 ms; single verify ≤ 50 ms; per-message OBU+RSU crypto cycle for the transmission of safety-critical Cooperative Awareness Messages (CAM): ≤ 250 ms, for the transmission of non-safety-critical Decentralized Environmental Notification Messages (DENM): ≤ 550 ms
Bandwidth	To check if proofs and certificates can fit within CAM and DENM message specifications. If they cannot, evaluate the key size that allows for the construction of signatures that do not fragment into more than 5 packets.
E2E crypto execution	Pure crypto E2E cycle ≤ 250 ms (excluding network overhead and latency); full-stack measurement on target network/hardware.
PQC computation overhead (cycles or time)	Per-message cycle overhead $\leq 10\times$ per session establishment (against the baseline of the case where traditional crypto is used).
Network Overhead	Encrypted CAM: $13\times$ size increase; exceeds 802.11p MTU (2,304 B); no issue on 4G/5G; maximum of 5 fragmentations allowed.
Number of mitigation strategies (PQ-TLS)	At least 4 mitigation strategies in total: 1 PQ-TLS, 1 PKI, and 2 OTA-related strategies (lattice-based / ML-DSA, FN-DSA).
Number of mitigation strategies (IKE)	≥ 2 , to capture two different KEMs, ML-KEM and HQC.
Hardware-based Acceleration	$5\times$ speedup, achievable via HQC and via NTT-based lattice operations with appropriate butterfly units ($16-17\times$).

Table 3.3: Acceptance criteria for the common evaluation metrics in the Automation use case (UC2)

Evaluation Metric	Acceptance Criteria
Latency	≤ 1000 ms additional overhead per handshake connection compared to the classical cryptography baseline.
Bandwidth	$\leq 20\%$ increase in message size overhead, targeting signature sizes < 1 KB through compression.

E2E crypto execution	Total process time (Sign → Transmit → Verify → Run) must be between 50 and 200 ms per task cycle.
PQC computation overhead (cycles or time)	SDK CPU overhead $\leq 10\%$ and Quantum-Resistant Trusted Computing Base (QR-TCB) $\leq 35\%$.
Network Overhead	$\leq 15\%$ increase in total bandwidth on live ROS 2 topics under PQ-TLS vs. plain DDS baseline.
Number of mitigation strategies (PQ-TLS)	≥ 2 strategies validated (Hybrid Key Exchange and dynamic fallback to classical algorithms).
Number of mitigation strategies (IKE)	N/A.
Hardware-based Acceleration	$\geq 15x$ speed-up for bottleneck operations (ML-KEM/ML-DSA) via FPGA/QR-TCB offloading.

Table 3.4: Acceptance criteria for the common evaluation metrics in the Finance use case (UC3)

Evaluation Metric	Acceptance Criteria
Latency	≤ 50 ms additional overhead per handshake connection compared to the classical cryptography baseline.
Bandwidth	Must sustain at least 90% of baseline transaction throughput capacity despite the integration of larger PQC key and signature sizes.
E2E crypto execution	The complete end-to-end cryptographic handshake and authentication process must conclude within < 200 ms.
PQC computation overhead (cycles or time)	≤ 35 ms additional processing time (or $\leq 10^7$ CPU cycles) per session establishment.
Network Overhead	$\leq 20\%$ increase in total packet size overhead during the connection setup phase (accounting for larger PQC certificates).
Number of mitigation strategies (PQ-TLS)	≥ 2 (e.g., implementation of Hybrid Key Exchange combining classical and PQC, and dynamic fallback to classical algorithms if PQC fails).
Number of mitigation strategies (IKE)	≥ 2 (e.g., support for Post-Quantum Pre-Shared Keys (PPK) and hybrid IKEv2 key exchanges).
Hardware-based Acceleration	Demonstrated support for FPGA/HSM hardware offloading for bottleneck operations (such as NTT or Karatsuba polynomial multiplications) to satisfy latency SLAs.

Table 3.5: Acceptance criteria for the common evaluation metrics in the Energy use case (UC4)

KPI	Acceptance Criteria
Secure Channel Establishment Time	Operational round-trip time required to negotiate and establish a mutually authenticated, post-quantum session (PQ-TLS using ML-KEM and ML-DSA / Falcon) between a gateway and the SCADA control center. ≤ 500 ms per secure session establishment handshake.
Telemetry Throughput Retention	Percentage of nominal real-time smart grid telemetry throughput retained post-PQC integration, relative to the classical legacy baseline. $\geq 95\%$ of telemetry data successfully transmitted within the designated time window (60 seconds).
Crypto-Agility Transition Time	Maximum runtime duration needed to dynamically hot-swap between supported PQC algorithm families (e.g., lattice ML-KEM-512 to ML-KEM-768, or code-based HQC) on an active gateway node without service interruption. ≤ 1 min total execution time per live algorithm migration cycle.
Authentication & Provisioning Failure Rate	Proportion of onboarding, initialization, or runtime authentication attempts that fail due to PQC-specific exceptions (e.g., lattice parameter rounding errors, key mismatches, or Falcon floating-point timing traps). $\leq 0.1\%$ of all cryptographic validation and verification attempts.
PQC Infrastructure Uptime	Global operational availability of the quantum-secured grid telemetry channels, the self-hosted PQaaS Backend proxy, and the local QR-TCB execution runtime. $\geq 99.5\%$ continuous availability during the scheduled pilot phase operations.

Crypto Inventory & Posture Coverage	Completeness of the discovery, cataloguing, and real-time posture monitoring of system primitives achieved by the Crypto Assessment Toolkit / Trust Assessment Framework (TAF). $\geq 90\%$ of active infrastructure components continuously mapped and audited for NIS2 compliance.
--	--

Table 3.6: Acceptance criteria for the common evaluation metrics in the Healthcare use case (UC5.1)

KPI	Description	Target / Acceptance Criteria
Latency	Overhead in latency introduced by integrating PQC mechanisms at the network and application layer of the edge device, measured via the PiQASO SDK. Covers all PQC operations integrated within the pilot, including secure channel establishment, encryption operations at the edge, cryptographic operations at the backend, and update signature verification. Latency is defined as the additional delay imposed by PQC mechanisms over the non-PQC baseline for each operation type (e.g., signature, data encryption) during runtime.	≤ 500 ms per PQC operation at the edge device. Pilot-specific acceptance thresholds are defined per operation in Section A above.
Bandwidth overhead	Additional bandwidth consumed by the introduction of PQC mechanisms, accounting for the increased size of post-quantum ciphertext payloads and PQ signatures introduced across the communication channels of the pilot. Larger payloads may cause fragmentation of messages, increasing the number of packets required per transmission. Measured in additional kilobytes per transmission to allow logical deduction of resulting packet overhead per communication channel.	≤ 50 KB additional bandwidth per transmission per PQC component, compared against the baseline deployment without PQC mechanisms. Results are reported per communication channel and per PQC primitive type.
E2E crypto execution	End-to-end cryptographic execution time covering the full PQC-protected data processing chain from data acquisition at the edge device through encryption, transmission, and encrypted storage at the backend. The acceptable threshold for this KPI is determined by the operational and safety profile of the deployment context rather than a generic cross-pilot value, and is therefore defined per pilot as an application-specific acceptance criterion.	≤ 2 s end-to-end cryptographic execution time, considering complexity of crypto abstractions (Functional Encryption, Attribute-Based Encryption) for capturing requirements. For the TelluCare pilot, a stricter application-specific threshold applies — see Section A, KPI <i>End-to-end cryptographic processing latency</i> (target: < 5 s, decomposed into FE and ABE components).
PQC computation overhead	Additional CPU overhead introduced by PQC mechanisms executing on the edge device via the PiQASO SDK, compared to the non-PQC baseline deployment. Covers all PQC operations integrated within the pilot, including secure channel establishment at onboarding, encryption operations for data confidentiality, cloud-side cryptographic key operations, and update package signature verification.	$\leq 15\%$ additional CPU overhead for PiQASO SDK integration compared to the non-PQC baseline. An additional $\leq 20\%$ overhead is permitted if Quantum-Resistant Trusted Computing Base (QRTCB) mechanisms are implemented at the edge device level.

Network overhead	Additional overhead introduced at the security protocol layer when migrating existing deployments of TLS, IPsec, or IKE to their post-quantum equivalents. This KPI applies only to pilots where such protocols are already in use in the baseline system and a direct protocol-level comparison between classical and post-quantum equivalents is therefore possible. In the TelluCare RPM pilot, the current onboarding process relies on a plain HTTP connection with no existing TLS or certificate exchange in place. The introduction of PQ-TLS therefore represents a new security capability rather than a migration of an existing protocol, and no protocol-level network overhead baseline exists for comparison.	N/A — The TelluCare baseline does not use TLS, IPsec, or IKE. PQ-TLS is introduced as a new security capability rather than a protocol migration; no protocol-level network overhead baseline exists for comparison.
Number of mitigation strategies	Number of distinct PQC-based security mechanisms and protocol configurations integrated and evaluated within the pilot. PQC primitives and security protocols are categorised under a unified mitigation strategies framework applicable across all PiQASO pilots, covering post-quantum secure channel mechanisms, encryption schemes, and signature schemes. The applicability of specific protocol categories such as IKE/IPsec is determined per pilot based on the baseline communication infrastructure.	4 mitigation strategies evaluated: (1) PQ-TLS for gateway onboarding and authentication, (2) Functional Encryption (FE) for medical data confidentiality, (3) Attribute-Based Encryption (ABE) for access-controlled cloud storage, and (4) PQ Signatures for OTA update integrity verification. IKE/IPsec: N/A.
Hardware-based acceleration	Reduction in PQC operation execution time achieved through hardware acceleration and algorithmic optimisation of PQC primitives on the hardware-accelerated edge platform, compared against equivalent software-only execution on the baseline platform. Evaluated across all PQC operations integrated within the pilot. Where hardware changes result in clock frequency reductions, any required partial software offloading must be accounted for in the reported speedup to ensure an accurate comparison.	Minimum 15× speedup in PQC operation execution time on the hardware-accelerated edge platform relative to the software-only baseline, sufficient to maintain the operational safety and data freshness profile of the deployment context.

Table 3.7: Acceptance criteria for the common evaluation metrics in the Healthcare use case (UC5.2)

Evaluation Metric	Acceptance Criteria
Latency	≤ 1000 ms additional overhead per handshake connection compared to the classical cryptography baseline.
Bandwidth	Increase payload size due to ML-DSA signature and ML-KEM encapsulation overhead does not exceed 20 KB to 50 KB per message, preventing packet fragmentation on standard Wi-Fi/cellular links.
E2E crypto execution	The total time elapsed from the sender side (PARTICLE GW) to the receiving side (PARTICLE server) must complete in ≤ 3.0 seconds under normal network loads.
PQC computation overhead (cycles or time)	The execution of local PQC software libraries must consume $\leq 5\%$ CPU average utilization and require ≤ 50 MB of peak memory (RAM).
Network Overhead	Capable to operate (100% packets delivery) over mobile networks that can operate as high-speed (broadband, e.g., 4G or 5G with throughput several Mbps) or limited networks (e.g., 3G with throughput order of Kbps). 100% recovery from periods of intermittent and loss of connectivity.
Number of mitigation strategies (PQ-TLS)	(1) PQ-TLS for establishing secure and authenticated channels; (2) local PQC SDK execution on the gateway (edge node); (3) FPGA QR-TCB-oriented hardware path with hardware acceleration for improved performance at the gateway (edge node). (4) Cryptographic agility: Fallback to classical algorithms in case destination hospitals do not support PQC algorithms.

Number of mitigation strategies (IKE)	N/A
Hardware-based Acceleration	more than 5x time reduction compared with baseline hardware (no HW acceleration)

Table 3.8: Acceptance criteria for the common evaluation metrics in the Aerospace use case (UC6.1)

Common Metric	Evaluation	Acceptance Criteria (UC6.1 – Aerospace)
Latency		PQC operations do not push the closed-loop response above the 100 ms aerospace bound (initial target).
Bandwidth		PQC key/signature and payload expansion fit the testbench link with no message loss at the nominal 20 Hz loop rate.
E2E crypto execution		End-to-end protected round trip ≤ 100 ms per cycle; one-time key agreement ≤ 150 ms.
PQC computation overhead (cycles or time)		Per-message AEAD $\leq$ a few ms; the heavier handshake cost is characterised in software and on the FPGA.
Network overhead		Overhead added by PQC framing is characterised; the loop stays within budget under nominal testbench network conditions.
Number of mitigation strategies (PQ-TLS)		N/A for UC6.1: payloads are secured at the application layer (AES-256-GCM over MQTT), not via PQ-TLS.
Number of mitigation strategies (IKE)		N/A for UC6.1: no IPsec/IKE is used.
Hardware-based acceleration		FPGA (270 MHz) speed-up of ML-KEM/ML-DSA/PQ-TLS over the software baseline is quantified and reported against the achievable loop rate.

Table 3.9: Acceptance criteria for the common evaluation metrics in the Aerospace use case (UC6.2)

Evaluation Metric	Acceptance Criteria
Latency	The end-to-end processing latency overhead introduced by PQC operations (ML-DSA authentication, key retrieval, and ML-KEM encapsulation) at the endpoint Cryptographic Module does not exceed an average of 1.5 seconds compared to the classical TLS baseline.
Bandwidth	The communication channels must successfully handle the increased payload sizes, ensuring that the combined ML-DSA signature and ML-KEM encapsulation overhead does not exceed 20 KB to 50 KB per message, preventing packet fragmentation on standard Wi-Fi/cellular links.
E2E crypto execution	The total time elapsed from the initial local encryption at the Sender's Cryptographic Module to the successful decapsulation, decryption, and RAD-signing at the PiQASO Gateway (Legacy Translation loop) must complete in ≤ 3.0 seconds under normal network loads.
PQC computation overhead (cycles or time)	The execution of local PQC software libraries must consume $\leq 5\%$ CPU utilization and require ≤ 50 MB of peak memory (RAM) during cryptographic bursts on standard aviation maintenance endpoints, ensuring zero disruption to concurrent applications.
Network Overhead	The transmission of the expanded packages over standard TLS maintains a 100% successful delivery rate with a packet drop rate of $\leq 1\%$, requiring no more than the standard 3 SMTP delivery retries.
Number of mitigation strategies (PQ-TLS)	At least 1 active mitigation strategy (e.g., fallback routing or session resumption) tested specifically on the internal PQ-TLS tunnel connecting the Internal Email Server to the PiQASO Gateway.
Number of mitigation strategies (IKE)	N/A - Architecture secures data at the application layer via ML-KEM encapsulation and at the transport layer, rather than network layer.
Hardware-based Acceleration	N/A - To meet requirements of the aviation maintenance sector, explicitly relies on software-only endpoint integration.

Table 3.10: Acceptance criteria for the common evaluation metrics in the Online Media use case (UC7)

Evaluation Metric	Acceptance Criteria
-------------------	---------------------

Latency	The end-to-end processing latency overhead introduced by PQC operations comprising the testing of the various PQ-TLS variants, including also KEMTLS-PDK variants than enhance server authentication based on pre-distributed shared keys, as well as the complex key management system for the creation and management of time-evolving key chains supporting the UPKE Attribute-based Encryption of data payloads. This should be less than 1sec ensuring it falls within the human perception threshold of “ <i>instantaneous</i> ”
Bandwidth	Communication rate of exchange data payloads (i.e., unpublished articles for review) should be in the order of Mgbps, independent of the size of exchanged client/server certification for authentication; The drop rate of such data payload packets should be kept less than 1%
E2E crypto execution	The entire process of user authentication; time epoch key establishment and update; data payload encryption/decryption; message integrity verification should not account for more than 45% of the overall time required for the secure upload and review of an unpublished document.
PQC computation overhead (cycles or time)	the elevation of the Content Management System to its PQC equivalent (supporting all necessary authentication, key management and signature operations) should not add a delta higher than 30% against the legacy TLS 1.3 protocol currently used. The percentage of TLS <i>ClientHello</i> or <i>ServerHello</i> messages that exceed the standard network Maximum Transmission Unit (MTU, typically 1500 bytes) and are split across multiple TCP segments should be kept at a minimum.
Network Overhead	Tracking if the large public credential size (PQ-encoded certificates) forces the connection into multiple Round-Trip times (RTTs) requiring from the user client to “wait” for out-of-order packets containing parts of the signature prior to the establishment of an authenticated encrypted communication channel. Packet fragmentation of certificate exchange ≤ 5 packets; RTT ≤ 45 ms
Number of mitigation strategies (PQ-TLS)	Evaluate interactive KEMTLS using the PiQASO PQC SW repository of lattice-based crypto, or code-based KEMs (ML-KEM, HQC, McEliece) for confidentiality and authentication. PQCaaS will also adopt KEMTLS-PDK variants that enhance server authentication based on pre-distributed shared keys, using McEliece for its short ciphertext size.
Number of mitigation strategies (IKE)	Evaluate the creation and maintenance of two types of time-evolving key chains: user-specific and attribute/policy-specific chains, supporting two modes of operation: i) Encryption/Decryption of User-Specific Data Payloads: Data sharing between a specific user (data receiver) and one or more senders utilizes the evolving user-specific key chains of PiQASO PQCaaS. ii) UPKE Attribute-Based Encryption of Data Payload: Data is shared between a specific sender and multiple receivers complying with a trusted third-party-defined policy.
Hardware-based Acceleration	Time required to reconfigure the PiQASO underlying HW (acceleration) module from an ML-KEM step to an ML-DSA verification step (≤ 660 ms)

Table 3.11: Acceptance criteria for the common evaluation metrics in the UAVs use case (UC8)

Evaluation Metric	Acceptance Criteria
Latency	Secure session establishment target: $p_{95} \leq 5$ s. Protected upload and in-flight route update shall remain within the operational timing limits defined in the UC8 validation plan.
Bandwidth	Authenticated telemetry delivery under nominal link conditions should be $\geq 90\%$ of the configured test rate, and the protected channel must not create persistent link saturation or repeated fragmentation failures.
E2E crypto execution	Protected upload and in-flight route-update success $\geq 90\%$ under nominal test conditions; route-update E2E $p_{95} \leq 5$ s after an established session; telemetry E2E overhead $\leq$ baseline + 1 s on 4G/5G. PQCaaS-assisted functions should complete with $\geq 90\%$ success under nominal back-haul.
PQC computation overhead (cycles or time)	The minimum acceleration target for hardware-accelerated primitives is $\geq 15\times$ speed-up compared with the PQ Clean software baseline. The validation shall compare edge computer execution, FPGA hardware-accelerated execution, and PQCaaS-assisted execution where applicable.

Network Overhead	Initial acceptance target: established-session communication overhead should remain within $\leq 40\text{--}60\%$ over the baseline for the agreed telemetry profile. One-off handshake, provisioning, certificate, or attestation bursts may exceed this range only if they do not violate the latency criteria and do not create persistent link saturation or repeated fragmentation failures. Authenticated telemetry delivery under nominal link conditions should remain $\geq 90\%$ of the configured baseline telemetry rate.
Number of mitigation strategies (PQ-TLS)	Evaluate at least three UC8 mitigation/configuration strategies: i) local PQC SDK execution on the edge computer secure node, ii) FPGA QR-TCB-oriented hardware path with hardware-acceleration benchmarking, and iii) PQCaaS-assisted non-hard-real-time support. If more than one secure-channel profile is available, compare at least two profiles; otherwise, document the single available profile.
Number of mitigation strategies (IKE)	N/A
Hardware-based Acceleration	The FPGA path satisfies secure-session, in-flight re-tasking, telemetry-delivery and key-isolation criteria; no long-term keys are released to the flight controller; no security-induced backlog occurs under nominal conditions; and the maximum usable acceleration is reported.

Table 3.12: Acceptance criteria for the common evaluation metrics in the Transportation use case (UC9)

Evaluation Metric	Acceptance Criteria
Latency	For the signing key generation, signature generation, signature verification, and key encapsulation using PQ-SDK $< 15ms$ compared to traditional operations using pycrypto. For the same operations using PQCaaS $< 50ms$ for a locally deployed PQCaaS instance
Bandwidth	$< 4Kb$ per transmitted anomaly report
E2E crypto execution	The operation that involves multiple crypto operations is anomaly report encryption. This involves key generation using ML-KEM, key derivation using HKDF, and AES-GCM encryption. The accepted E2E time is $< 10ms$ per anomaly report. Similar delay is expected for the decryption process
PQC computation overhead (cycles or time)	$< 15ms$ increase compared to the execution of the corresponding traditional algorithms
Network Overhead	$\leq 35\%$ increase of the consumed bandwidth for completing an operation compare to transmitting an anomaly signed using traditional cryptography over TLS
Number of mitigation strategies (PQ-TLS)	≥ 3 The use case will consider PQ-TLS, ML-KEM, and ML-DSA
Number of mitigation strategies (IKE)	N/A
Hardware-based Acceleration	$> 10x$ faster cryptographic operations compared to software-only implementations

3.4 Common PiQASO Operation Sequence Diagrams

As detailed throughout this Chapter, one core outcome of the PiQASO methodology was *the identification of the PiQASO components and functionalities which are relevant for the post-quantum transition of large-scale heterogeneous service graph chains, regardless of application or industrial domain*. To this end, here we provide a set of common sequence diagrams corresponding to these identified operations, namely (i) two variants of **Post-Quantum TLS (PQ-TLS)** for **direct client-server communication** and **authentication via the PQCaaS server**, the (ii) **Updatable Public Key Encryption (UPKE)**, and the (iii) **Attribute-Based Encryption (ABE)**.

3.4.1 PQ-TLS - Variant 1: Direct Client-Server Authentication

As aforementioned, PiQASO supports two variants of PQ-TLS, namely direct client-server authentication and authentication through PQCaaS for devices that do not have the capability to support the required mathematical operations. This Section is dedicated to the first variant, depicted in Figure 3.1. In the following, we provide a step-by-step description of the sequence of actions comprising this protocol. Note that this protocol is quite similar to the traditional TLS protocol, and the PiQASO scheme mainly pertains to introducing post-quantum crypto in the PQ-TLS Handshake (Phase 2).

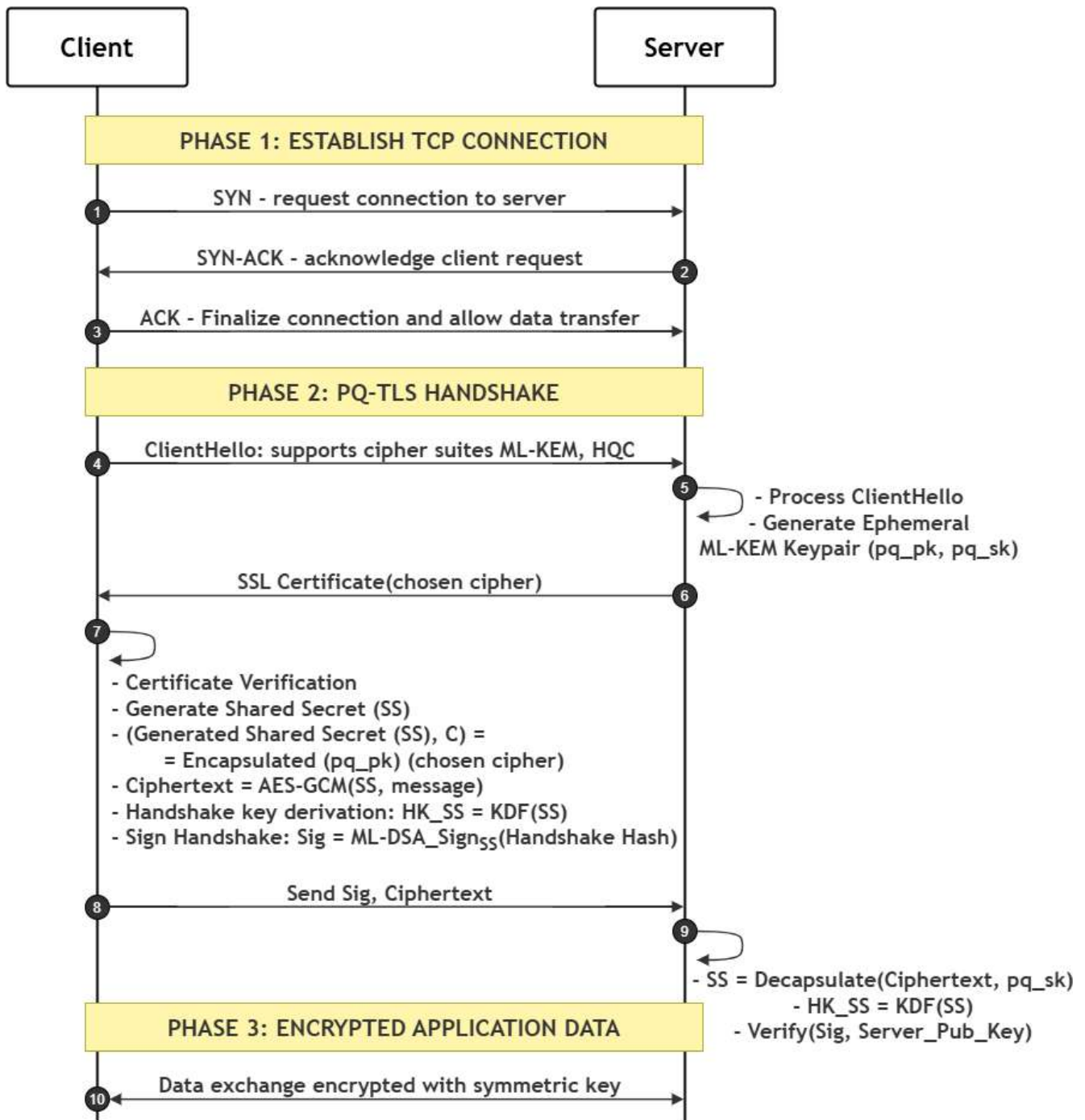


Figure 3.1: PiQASO PQ-TLS - Variant 1: Direct Client-Server Authentication

1. **SYN:** The Client sends a SYN packet to request a connection to the Server.
2. **SYN-ACK:** The Server acknowledges the client's request with a SYN-ACK packet.

3. **ACK:** The Client finalizes the TCP connection with an ACK packet, allowing data transfer to begin. Note that, up until this step, this process is identical to the traditional TLS protocol. Starting with the next step, the PiQASO PQ-TLS protocol is differentiated through the introduction of PQC for the TLS handshake.
4. **ClientHello:** The Client sends a ClientHello message indicating support for the ML-KEM and HQC post-quantum cipher suites.
5. **Server-side Processing:** The Server processes the ClientHello message and generates an ephemeral ML-KEM keypair (public key pq_pk and secret key pq_sk).
6. **SSL Certificate:** The Server sends its SSL Certificate to the Client, including the chosen cipher suite.
7. **Client-side Cryptographic Operations:** The Client performs the following steps:
 - (a) Verifies the Server's certificate.
 - (b) Generates a Shared Secret (SS) via encapsulation: $(SS, C) = \text{Encapsulate}(pq_pk)$ using the chosen cipher.
 - (c) Encrypts a message using the shared secret: $\text{Ciphertext} = \text{AES-GCM}(SS, \text{message})$.
 - (d) Derives the handshake key: $HK_SS = \text{KDF}(SS)$.
 - (e) Signs the handshake hash using ML-DSA: $\text{Sig} = \text{ML-DSA_Sign}_{sk}(\text{Handshake Hash})$.
8. **Send Signature and Ciphertext:** The Client transmits the signature (Sig) and the ciphertext (C) to the Server.
9. **Server-side Verification:** The Server performs the following steps:
 - (a) Decapsulates the ciphertext using its secret key: $SS = \text{Decapsulate}(\text{Ciphertext}, pq_sk)$.
 - (b) Derives the handshake key: $HK_SS = \text{KDF}(SS)$.
 - (c) Verifies the client's signature: $\text{Verify}(\text{Sig}, \text{Server_Pub_Key})$.
10. **Encrypted Data Exchange:** Both parties are now able to exchange application data encrypted with the derived symmetric key.

3.4.2 PQ-TLS - Variant 2: Authentication Through PQCaaS

This section is dedicated to the description of the action workflow of the second variant of the PiQASO PQ-TLS protocol, depicted in Figure 3.2, which is intended to provide PQ-TLS capabilities to Senders that do not have the capability to support the required complex mathematical operations. Note that, unlike Variant 1, where the Client and Server perform the PQ key exchange directly, Variant 2 splits trust across two independently-secured legs (steps 8–9 and 11–13) that are stitched together at the session layer by nginx-pqc (step 15). The security of the overall channel therefore depends on the PQCaaS Server being a trusted intermediary, since it is the only party that holds both `cert_1` and `cert_2` in plaintext at any point.

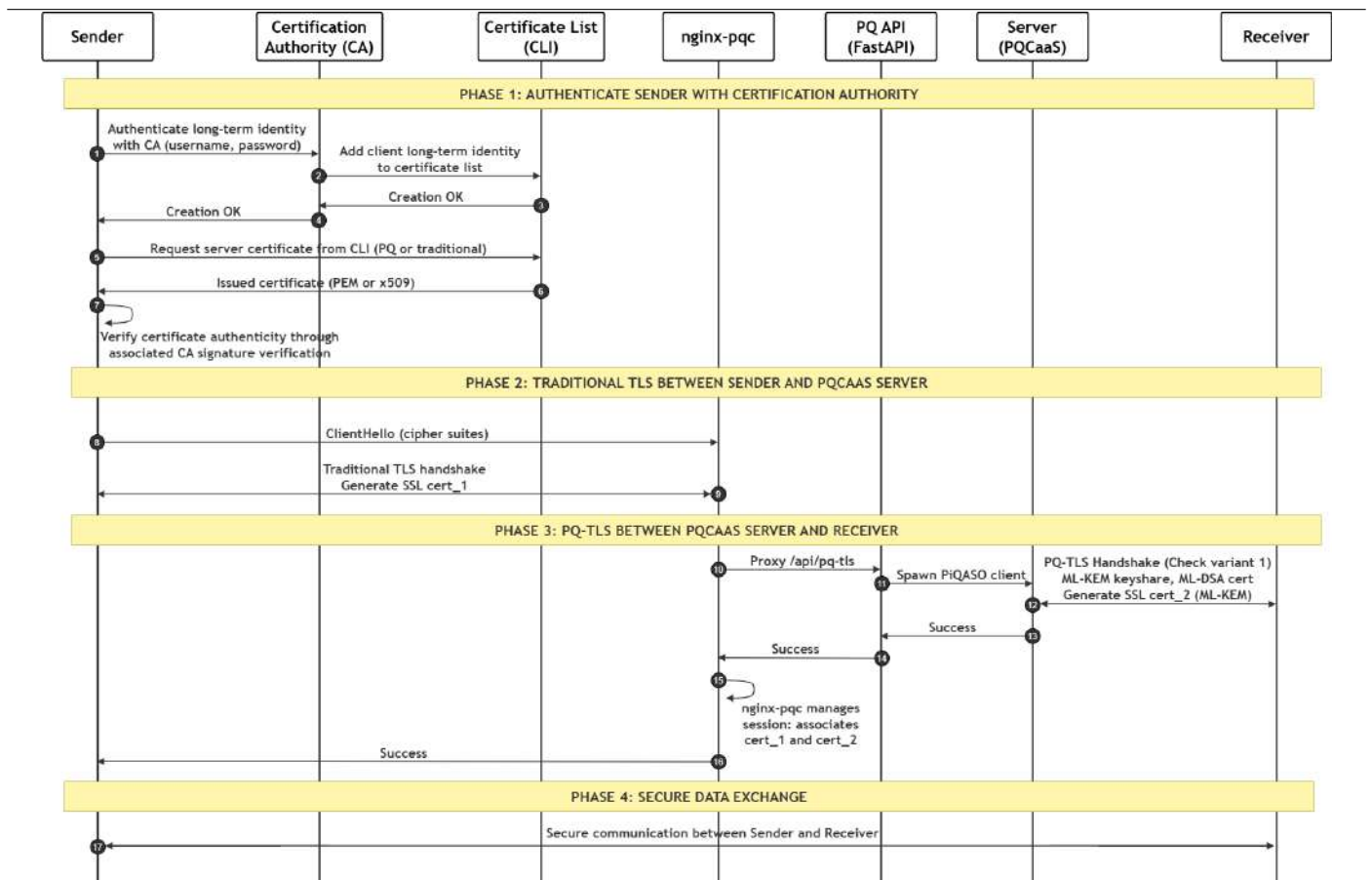


Figure 3.2: PiQASO PQ-TLS - Variant 2: Authentication through PQCaaS

1. **Authenticate Long-Term Identity:** The Sender authenticates its long-term identity with the Certification Authority (CA) by submitting credentials (username, password).
2. **Register Identity:** The CA adds the Sender's verified long-term identity to the Certificate List (CLI).
3. **Confirm Registration:** The CLI returns a "Creation OK" confirmation to the CA, acknowledging that the identity has been recorded.
4. **Relay Confirmation:** The CA forwards the "Creation OK" confirmation to the Sender, completing the identity registration step.
5. **Request Server Certificate:** The Sender requests a server certificate from the CLI, specifying whether a post-quantum (PQ) or traditional certificate is required.
6. **Issue Certificate:** The CLI issues the requested certificate to the Sender in PEM or X.509 format.
7. **Verify Certificate Authenticity:** The Sender verifies the authenticity of the issued certificate through the associated CA signature.
8. **ClientHello (Traditional):** The Sender initiates a traditional TLS handshake by sending a ClientHello message, listing supported (classical) cipher suites, to the PQCaaS Server.
9. **Traditional Handshake Completion:** The Sender and the PQCaaS Server complete a traditional TLS handshake, during which the Server generates an SSL certificate (cert_1) to secure this leg of the connection. This establishes a classically-secured channel between the Sender and the PQCaaS Server, which will act as an intermediary on the Sender's behalf for the post-quantum portion of the exchange.

10. **Proxy PQ-TLS Request:** The nginx-pqc reverse proxy receives the request and forwards it to the PQ API (FastAPI) via the `/api/pq-tls` endpoint, initiating the post-quantum handshake process on behalf of the Sender.
11. **Spawn PiQASO Client:** The PQ API spawns a PiQASO client process, which will perform the actual PQ-TLS handshake with the Receiver, following the logic of PQ-TLS Variant 1 (see corresponding diagram).
12. **PQ-TLS Handshake Execution:** The spawned PiQASO client and the Receiver execute the PQ-TLS handshake (per Variant 1): exchanging the ML-KEM keyshare and ML-DSA certificate, and the Receiver generates a second SSL certificate (`cert_2`) based on ML-KEM key encapsulation.
13. **Handshake Success (Receiver):** The Receiver confirms successful completion of the PQ-TLS handshake to the PQCaaS Server.
14. **Handshake Success (PQ API):** The PQCaaS Server relays the success confirmation back to the PQ API.
15. **Session Association:** The nginx-pqc proxy manages the overall session state, internally associating the traditional certificate (`cert_1`, from Phase 2) with the post-quantum certificate (`cert_2`, from Phase 3), effectively bridging the Sender's classical session to the Receiver's PQ session.
16. **Confirm Success to Sender:** The nginx-pqc proxy returns a final success confirmation to the Sender, indicating that the full Sender-to-Receiver secure channel (via the PQCaaS intermediary) has been established.
17. **Secure Data Exchange:** The Sender and Receiver exchange data securely. Although the two parties never establish a direct PQ-TLS session with one another, secure communication is achieved end-to-end. Specifically, the Sender-PQCaaS leg uses traditional TLS (`cert_1`), and the PQCaaS-Receiver leg uses PQ-TLS (`cert_2`), with the PQCaaS Server acting as a trusted intermediary that bridges the two cryptographic domains. This design allows a Sender lacking native post-quantum computation capabilities to still benefit from PQ-secured communication with the Receiver.

3.4.3 Updatable Public Key Encryption

This section is dedicated to the PiQASO **Updatable Public Key Encryption (UPKE)** scheme, depicted in Figure 3.3, whose purpose is to enable secure key update functionalities to protect data at rest, while ensuring that forward secrecy is guaranteed. In the following, we provide a detailed step-by-step description of the scheme. Note that the defining UPKE property is visible in steps 6 and 16, where it is shown that the PQCaaS Server can refresh the public key/epoch parameter associated with a Receiver's private key (thus achieving forward security against future key compromise).

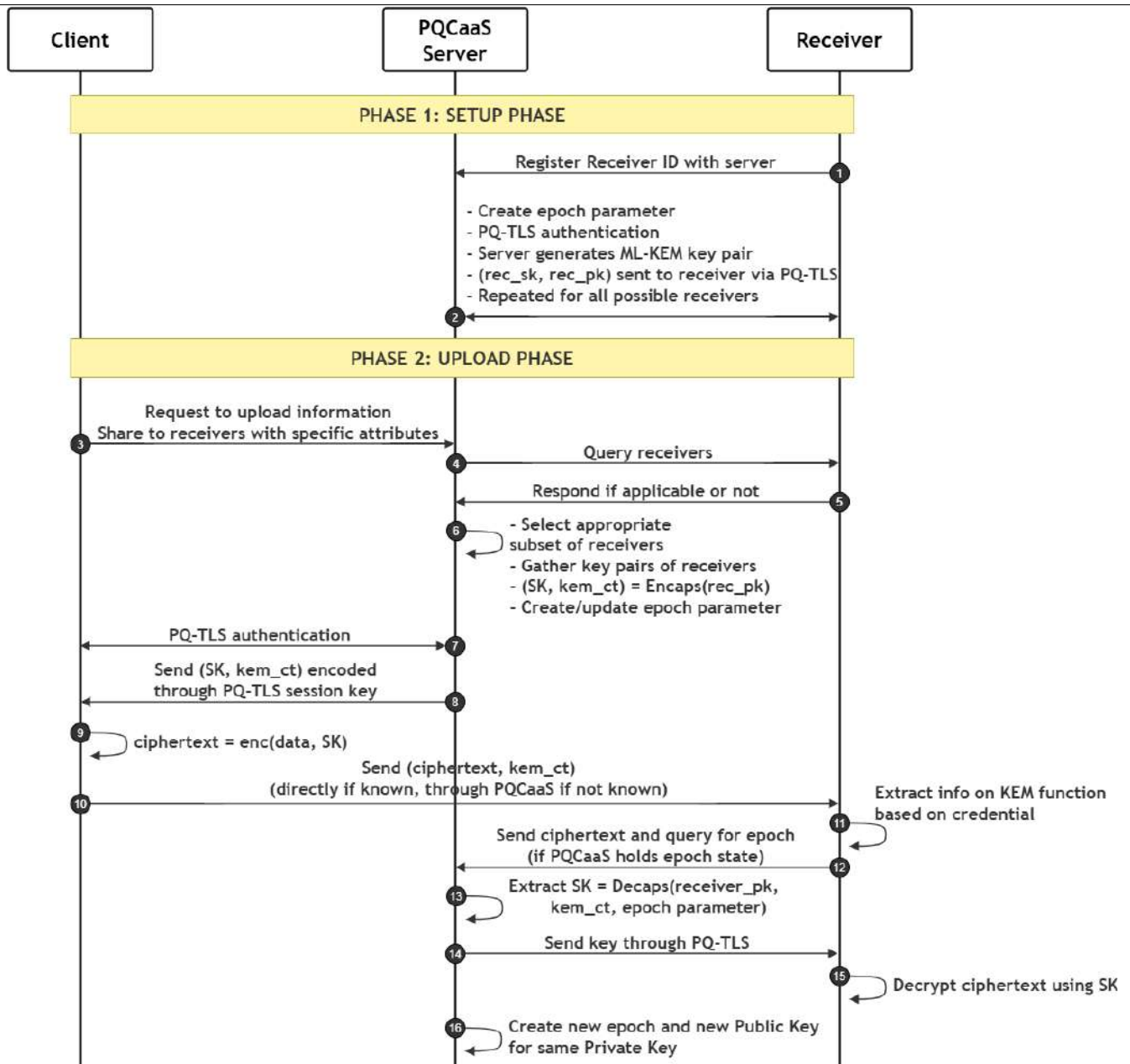


Figure 3.3: PiQASO Updatable Public Key Encryption

1. **Register Receiver ID:** The Receiver registers its identity with the PQaaS Server, establishing itself as an eligible recipient within the system.
2. **Setup and Key Provisioning:** The PQaaS Server performs the initial setup for this Receiver:
 - (a) Creates an epoch parameter, marking the current key-versioning state for this Receiver.
 - (b) Authenticates the channel using PQ-TLS.
 - (c) Generates an ML-KEM key pair for the Receiver.
 - (d) Sends the private/public key pair (rec_sk, rec_pk) to the Receiver over the authenticated PQ-TLS channel.
 - (e) Repeats this entire process for every possible Receiver registered with the system, so that the Server holds key material for the full receiver population.
3. **Upload Request:** The Client requests to upload information, specifying that it should be shared with receivers matching certain attributes (i.e., attribute-based targeting of the ciphertext).

4. **Query Receivers:** The PQCaaS Server queries the relevant Receiver(s) to determine eligibility against the requested attributes.
5. **Eligibility Response:** The Receiver responds, indicating whether it is applicable, given the requested attributes.
6. **Receiver Subset Selection and Key Encapsulation:** The PQCaaS Server performs the following:
 - (a) Selects the appropriate subset of eligible Receivers.
 - (b) Gathers the corresponding public key pairs (rec_pk) for the selected subset.
 - (c) Generates a shared symmetric key and its encapsulation: $(\text{SK}, \text{kem_ct}) = \text{Encaps}(\text{rec_pk})$.
 - (d) Creates or updates the epoch parameter associated with this key encapsulation, tying it to the current key-update cycle.
7. **Client Authentication:** The Client and PQCaaS Server establish a PQ-TLS authenticated session.
8. **Key Material Delivery:** The Server sends the symmetric key and its encapsulation ($\text{SK}, \text{kem_ct}$) to the Client, encoded and protected under the PQ-TLS session key.
9. **Local Encryption:** The Client encrypts its data locally using the received symmetric key: $\text{ciphertext} = \text{enc}(\text{data}, \text{SK})$.
10. **Ciphertext Transmission:** The Client sends the pair ($\text{ciphertext}, \text{kem_ct}$) to the Receiver — directly if the Receiver's address/route is known, or relayed through the PQCaaS Server otherwise.
11. **Credential-Based KEM Lookup:** The Receiver extracts the information needed to identify the correct KEM function/parameters based on its own credential context (self-processing step).
12. **Epoch Query:** The Receiver sends the ciphertext to the PQCaaS Server along with a query for the relevant epoch state, applicable when the Server is the holder of epoch state rather than the Receiver.
13. **Shared Secret Recovery:** The Receiver (or the Server, depending on where epoch state resides) extracts the symmetric key by decapsulating: $\text{SK} = \text{Decaps}(\text{receiver_pk}, \text{kem_ct}, \text{epoch parameter})$.
14. **Key Delivery:** The recovered key is sent to the Receiver over a PQ-TLS protected channel (relevant when decapsulation was performed server-side).
15. **Data Decryption:** The Receiver decrypts the received ciphertext using the recovered symmetric key SK.
16. **Key Update (Epoch Rotation):** The PQCaaS Server creates a new epoch and a corresponding new public key for the same underlying private key, completing the “updatable” property of the scheme: future encapsulations will use the rotated public key/epoch, while the Receiver's long-term private key remains unchanged. This step provides forward-looking key freshness without requiring re-distribution of new private key material to the Receiver.

3.4.4 Attribute-Based Encryption

This section is dedicated to the **Attribute-Based Encryption (ABE)** scheme of PiQASO, whose purpose is to provide the capability for encrypting data in a manner that only allows their decryption by an entity that can verifiably demonstrate the possession of a particular attribute design space. The sequence

diagram for this scheme is depicted in Figure 3.4, and here we provide the corresponding detailed action workflow.

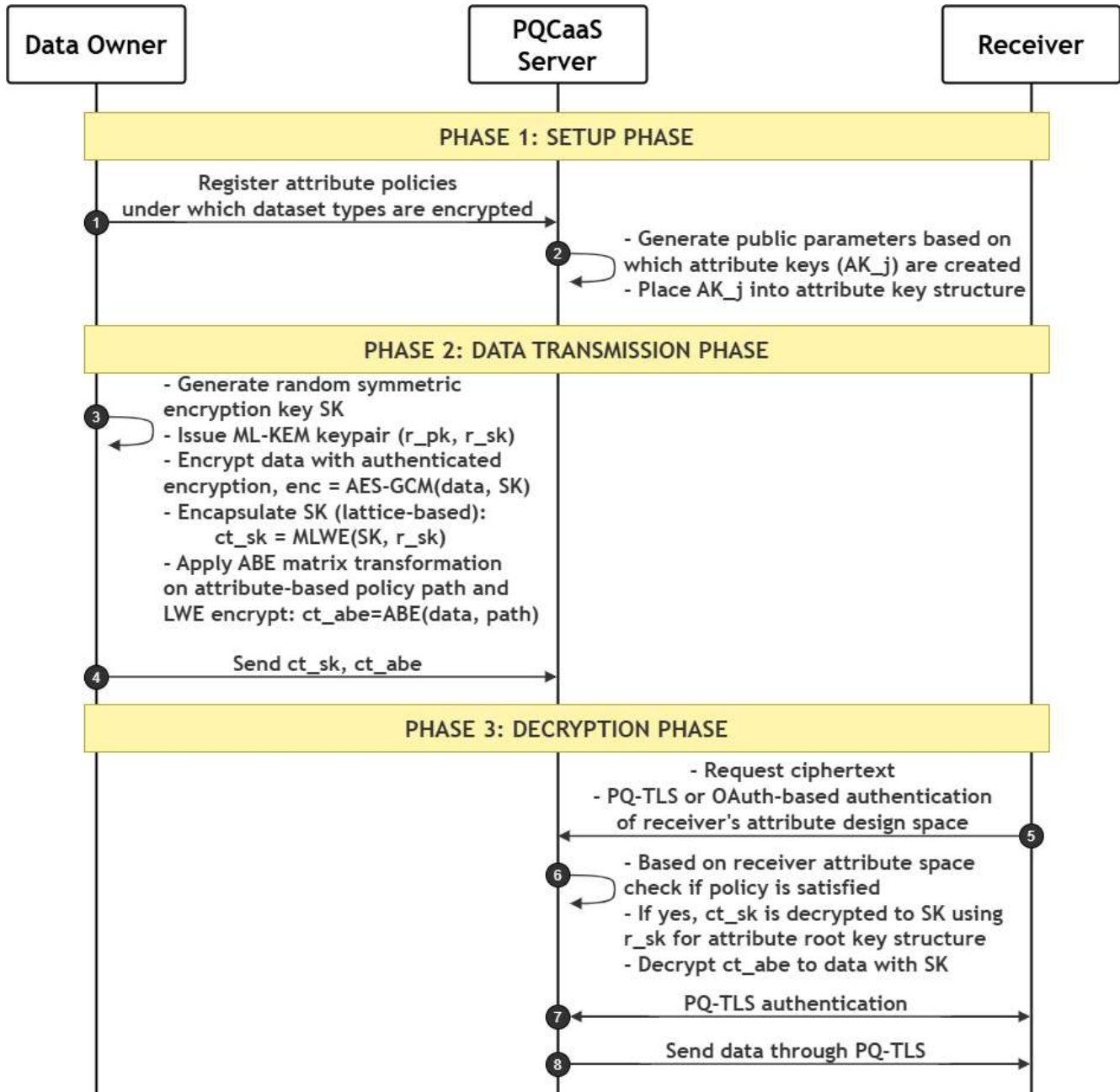


Figure 3.4: PiQASO Attribute-Based Encryption (ABE)

1. **Register Attribute Policies:** The Data Owner registers, with the PQcaaS Server, the attribute-based policies under which each dataset type should be encrypted (i.e., which combinations of attributes should be authorized to decrypt which data).
2. **Public Parameter Generation:** The PQcaaS Server processes the registered policies:
 - (a) Generates public parameters from which individual attribute keys (AK_j) can be created.
 - (b) Places each generated attribute key AK_j into an attribute key structure, organizing keys according to the policy hierarchy.
3. **Local Key Generation and Encryption (Data Owner):** The Data Owner performs the following sequence locally, prior to transmission:

-
- (a) Generates a random symmetric encryption key SK.
 - (b) Issues an ML-KEM key pair (r_{pk} , r_{sk}).
 - (c) Encrypts the data under authenticated encryption: $enc = AES-GCM(data, SK)$.
 - (d) Encapsulates the symmetric key using a lattice-based (MLWE) mechanism: $ct_{sk} = MLWE(SK, r_{sk})$.
 - (e) Applies the ABE matrix transformation along the attribute-based policy path and performs LWE encryption: $ct_{abe} = ABE(data, path)$.
 4. **Ciphertext Upload:** The Data Owner sends both ciphertext components, ct_{sk} and ct_{abe} , to the PQCaaS Server for storage and later distribution.
 5. **Ciphertext Request and Authentication:** The Receiver requests the ciphertext from the PQCaaS Server, authenticating its attribute credentials via PQ-TLS or OAuth-based authentication, which establishes the Receiver's attribute design space (i.e., which attributes it legitimately holds).
 6. **Policy Evaluation and Decryption:** The PQCaaS Server performs the following on behalf of the Receiver:
 - (a) Checks whether the Receiver's authenticated attribute space satisfies the policy associated with the requested data.
 - (b) If the policy is satisfied, decrypts ct_{sk} to recover SK, using r_{sk} together with the attribute root key structure (i.e., the attribute keys AK_j corresponding to the Receiver's satisfied attribute path).
 - (c) Decrypts ct_{abe} using the recovered SK to obtain the underlying plaintext data.
 7. **Session Authentication:** The PQCaaS Server and Receiver establish a PQ-TLS authenticated session for secure delivery.
 8. **Data Delivery:** The decrypted data is sent to the Receiver over the PQ-TLS protected channel. Thus, the Receiver whose attributes have been shown to satisfy the Data Owner's original policy is able to obtain access to the decrypted set of data.

Chapter 4

Cryptographic Asset Inventory Instantiation and Use-Case Specific SBOM

The PiQASO **Crypto Asset Inventorying (CAI)** component provides a foundational observability, discovery, and evidence-collection capability within the PIQASO reference architecture. Its purpose is *to build a structured, evidence-based understanding of the cryptographic and software-dependency posture of an operational environment before any post-quantum protection mechanisms are introduced*. CAI therefore serves as the entry point of the post-quantum migration workflow: it determines **where cryptography is used, how it is instantiated, which dependencies constrain migration, and which assets, communication flows, or services should be prioritised for quantum-resistant protection**.

Such a capability is most needed in today's **mixed-criticality, heterogeneous service graph chains**, which tend to combine long operational lifecycles, varied hardware and software baselines, legacy communication protocols, constrained tactical nodes, supplier-dependent subsystems, sensitive data flows, and demanding continuity-of-operations requirements. Cryptography is commonly utilized in several layers of the stack, such as *secure boot chains, firmware update procedures, authentication services, SSH keys, TLS channels, VPN configurations, package dependencies, software libraries, certificates, and applications*. Because of this, the cryptographic exposure of a service graph chain **cannot be derived from documentation alone**; it must be uncovered through **systematic inspection, runtime observation, and structured evidence collection**.

This direction matches the wider international consensus on post-quantum migration. The **NIST National Cybersecurity Center of Excellence (NCCoE)** migration programme emphasises that organisations must first understand where **quantum-vulnerable public-key algorithms** are deployed across hardware, software, and services before they can draw up prioritised migration roadmaps [48]. Likewise, **OMB Memorandum M-23-02** names the preparation of *prioritised inventories of cryptographic systems* as a preparatory step for the move to post-quantum cryptography [52]. Together, these positions affirm a key architectural principle: **post-quantum migration is not just an algorithmic substitution, but a system-level transition that starts with cryptographic visibility**.

In this setting, **CAI is not an enforcement component**. It *neither replaces cryptographic protocols, nor makes access-control decisions, nor stands up post-quantum services by itself*. What it provides is **the evidence base needed for informed migration and risk treatment**. Its outputs inform *crypto-profile selection, migration prioritisation, conformity assessment, system hardening, SBOM/CBOM enrichment*, and the choice of suitable PIQASO protection modalities. The separation of roles is deliberate: **CAI discovers and characterises the cryptographic posture**, while other PIQASO components supply the matching protection, acceleration, trusted execution, service-based offloading, logging, or path-establishment functions.

CAI spans several complementary observation layers. At the **network layer**, it inspects live or recorded

TLS traffic to determine *negotiated cipher suites*, *key exchange mechanisms*, and how far communication flows still rely on classical public-key cryptography. At the **host and software-dependency layer**, it examines installed packages and dependencies to flag *insecure*, *deprecated*, or *migration-relevant components* that could affect the system's cryptographic assurance. At the **authentication-material layer**, it reviews SSH key material and captures metadata such as *ownership*, *key type*, and *key length*, helping to pinpoint authentication assets that may need rotation, replacement, or tighter policy control. At the **software-analysis layer**, it examines local source-code repositories to find cryptographic libraries and functions, adding source-level insight into embedded cryptographic dependencies alongside the network-level view.

At implementation level, the first CAI release has been made available as a project software artefact through the **PIQASO GitLab repository**: https://gitlab.com/piqaso/crypto_asset_inventory [57]. The repository acts as the reference location for the initial CAI implementation and its subsequent refinements. This first release is organised around **four analysis modes** that correspond to distinct sources of cryptographic evidence.

- The **Network Analysis mode (NWA)** observes live TLS traffic or processes packet-capture files in order to extract *negotiated cipher suites*, *key exchange mechanisms*, and indicators of classical, hybrid, or post-quantum communication behaviour.
- The **Package Manager Analysis mode (PMA)** inspects host-level package information and dependency relationships, with the objective of identifying software components that may introduce security exposure or constrain cryptographic migration.
- The **SSH Key Analysis mode (SSHA)** enumerates SSH private-key material under user directories and records *key type*, *key length*, *ownership*, and *file location*, thereby supporting authentication-material inventorying.
- The **Software Analysis mode (SFA)** scans local source-code repositories and identifies the use of cryptographic libraries or crypto-related functions, providing static evidence of embedded cryptographic dependencies.

Table 4.1: CAI analysis modes and architectural role

CAI Mode	Observation Source	Evidence Produced	Architectural Role	Reasoning
nwa	Live TLS traffic or packet-capture files	Negotiated cipher suites, key exchange mechanisms, post-quantum readiness indicators	Identifies communication flows requiring PQ-TLS, hybrid TLS, or other post-quantum channel-protection mechanisms	As the following results show, the majority of use cases rely on TLS for authentication. Following discussions with the use-case partners, TLS was confirmed as the prevailing transport-security mechanism. The observation source is therefore set to TLS traffic and the tool is tailored accordingly.
pma	Host package database and dependency information	Installed packages, dependency relationships, insecure or migration-relevant components	Supports host-level exposure analysis and software-dependency visibility for migration planning	Enumerates the cryptographic libraries present on the operating system irrespective of whether they are actively used: their mere presence on the host constitutes a valid source of cryptographic evidence.

ssh	SSH key material under user directories	Key type, key length, owner, and file location	Supports authentication-material inventorying and identification of keys requiring rotation or future quantum-resistant replacement	Included for completeness. It has not been exercised in the current use cases but is retained as a capability of the tool.
sfa	Local source-code repositories	Cryptographic libraries, crypto-related functions, and source-code locations	Detects embedded cryptographic dependencies and informs software-level integration with PIQASO PQC components	Identifies which cryptographic libraries are invoked, whether for the handshake or for authenticated key establishment. The aim is not merely to detect a library's presence, but to capture the context in which each library is used.

The CAI processing workflow can be decomposed into **five logical steps**:

1. The operator or integration team selects the appropriate observation scope, such as a communication flow, packet-capture file, host package database, SSH key directory, or software repository.
2. The relevant CAI module extracts raw evidence from the selected source.
3. The extracted evidence is normalised into **structured JSON reports**, ensuring that findings can be further processed by migration, assessment, or documentation workflows.
4. The findings are interpreted in terms of cryptographic posture, including the identification of *classical public-key mechanisms, hybrid or post-quantum indicators, weak or deprecated algorithms, vulnerable software dependencies, and authentication material requiring policy review*.
5. The resulting evidence is mapped to **SBOM/CBOM-relevant records** and used to derive migration-relevant actions.

In runtime communication analysis, CAI rates the observed TLS-related cryptographic behaviour by **post-quantum readiness**. A flow can be judged **post-quantum ready** when post-quantum key establishment is seen, **acceptable or transitional** when hybrid classical/post-quantum mechanisms are present, **partial** when only certain layers of the cryptographic stack show quantum-resilient traits, or **vulnerable** when it still depends on classical public-key cryptography. This rating is not a replacement for a full security evaluation of the protocol deployment, but it gives an actionable first signal of where post-quantum migration should be prioritised.

In host and software analysis, CAI targets the cryptographic dependencies that frequently sit hidden below the operational service layer. **Package-level analysis** reveals installed components and dependency chains that can shape the feasibility of bringing in PQC-enabled libraries or services. **SSH-key analysis** reveals classical authentication material that may call for rotation, tighter lifecycle management, or future quantum-resistant replacement. **Source-code analysis** rounds out the runtime and host-level views by showing where cryptographic functions or libraries are invoked directly in software artefacts, which is a point of particular importance for *mixed-criticality, heterogeneous service graph chains*, where cryptographic logic is often buried in software, middleware, maintenance scripts, or integration components rather than isolated in dedicated security appliances.

The evidence collected by CAI contributes to the generation and enrichment of both **Software Bills of Materials (SBOM)** and **Cryptographic Bills of Materials (CBOM)**. The SBOM view captures the software supply-chain dimension of the assessed environment, including packages, libraries, dependencies, and software components. CISA describes SBOMs as a key building block in software security and software supply-chain risk management, enabling organisations to better understand and manage software-related risk [19]. The CBOM view extends this principle to cryptographic assets, capturing algorithms,

keys, certificates, protocols, cipher suites, cryptographic libraries, and their relationship to software components. CycloneDX positions CBOMs as a means to provide transparency into cryptographic assets and to assess weaknesses such as deprecated algorithms or expired certificates [54].

The **distinction between SBOM and CBOM** is essential for the post-quantum transition of the types of organizations examined in the context of PiQASO. **An SBOM may reveal that a system depends on a given software package or library**, but this does not necessarily explain *how cryptography is used by that component, which keys or certificates are involved, whether classical key exchange is negotiated at runtime, or whether a given protocol configuration is quantum-vulnerable*. Conversely, **a CBOM captures cryptographic objects and their operational relationships**, but it must remain connected to the software, service, or platform context in which these objects are instantiated. PIQASO therefore treats SBOM and CBOM as **complementary views of the same operational reality**: the former supports software-dependency and supply-chain visibility, while the latter supports cryptographic exposure analysis and post-quantum migration planning.

Architecturally, CAI follows an **evidence-to-action workflow**. It begins by collecting observations from selected infrastructure elements, i.e., endpoints, repositories, traffic captures, package databases, and user authentication directories. It then normalises the collected information into structured reports, and maps the discovered cryptographic and software artifacts onto SBOM/CBOM-relevant records. Then it classifies the identified assets by *cryptographic posture, quantum exposure, migration relevance, operational criticality, and dependency context*. Lastly, it supplies input to the PIQASO migration workflow, allowing suitable post-quantum protection mechanisms to be chosen for the identified assets, services, and communication flows.

This workflow is what enables **risk-based migration**. Communication flows that depend on classical key exchange can be prioritised for PQ-TLS or hybrid TLS integration. SSH keys built on classical algorithms can be earmarked for rotation planning and eventual quantum-resistant replacement. Software components relying on classical cryptographic libraries become candidates for integration with the **PIQASO PQC SDK**, which offers *ML-KEM, ML-DSA, LMS, XMSS, and AES* primitives over a WolfSSL backend. Hosts or embedded platforms unable to run computationally intensive post-quantum operations locally can be directed to **PQC-as-a-Service** or to hardware-accelerated offloading. Components that hold long-term keys or perform sensitive cryptographic operations can be allocated to trusted execution inside an **OP-TEE-based quantum-resistant trusted computing base (QR-TCB)**. Communication paths that cross several infrastructure nodes may need attestation, QR-Ledger support, and the establishment of a trusted path.

CAI also contributes to **crypto-agility**. Cryptographic agility requires more than the ability to configure a new algorithm. It requires continuous knowledge of *which algorithms are used, which protocols depend on them, which certificates and keys are affected, which libraries must be upgraded, and which operational services may be disrupted by a transition*. NIST defines cryptographic agility as **the ability to replace and adapt cryptographic algorithms across protocols, applications, software, hardware, firmware, and infrastructures while preserving security and ongoing operations** [13]. In this sense, CAI provides the visibility layer upon which crypto-agility can become an operational property rather than an abstract design principle.

The relevance of CAI is further strengthened by the need to **avoid uncontrolled migration**. In operational environments, especially those involving command-and-control systems, tactical communication links, UAV platforms, and 5G-enabled infrastructures, the direct replacement of cryptographic mechanisms without prior inventorying may introduce unacceptable risks. These risks include *interoperability failures, loss of legacy connectivity, unexpected latency, certificate-chain incompatibilities, insecure fallback behaviour, and incomplete protection of critical flows*. CAI reduces these risks by enabling **staged migration**, where the existing cryptographic posture is first understood, then prioritised, and finally transformed through suitable PIQASO protection mechanisms.

The outputs of CAI are thus meant to inform both technical and operational decisions. At a minimum, a CAI report should state *the asset or flow assessed, the observation source, the detected protocol or software component, the cryptographic algorithm or library in use, any available key or certificate characteristics, the relevant configuration context, the assessed quantum exposure, and a recommended migration or mitigation action*. System integrators, security administrators, use-case operators, and conformity-assessment activities can draw on this evidence to decide whether an asset should stay temporarily in a classical mode, move to a hybrid mode, migrate to a post-quantum profile, be protected through trusted execution, or be served through service-based cryptographic offloading.

CAI consequently sits **upstream of the remaining PIQASO architectural enablers**. Signals where the PIQASO PQC SDK should be integrated, where hardware acceleration may be needed, where QR-TCB protection is warranted, where PQC-as-a-Service is operationally suitable, where PQ-TLS or PQ-IPsec profiles should be brought in, and where QR-Ledger or trusted path mechanisms can provide auditability and trust continuity. This upstream placement ensures that the PIQASO mechanisms are applied not in a blanket fashion but in line with the actual cryptographic exposure, operational role, and resource profile of each asset.

As the following chapters show, **CAI is instantiated as the first step in every PIQASO use case**. This is a deliberate architectural decision: each use case must first fix its *as-is* cryptographic and software-dependency baseline before the matching *to-be* quantum-resistant protection mechanisms can be introduced. Crypto asset inventorying therefore becomes the **operational entry point for PQC migration**, enabling evidence-driven hardening of modern service graph chains and supporting a controlled transition toward quantum-resistant communication, authentication, trusted execution, and secure data-sharing capabilities.

With this in mind, the next chapter presents the crypto asset inventory information collected from the PIQASO partners up to this point. Produced by running the CAI analysis modes across the partner environments, these early submissions are summarized in a set of tables that give a first view of the cryptographic and software-dependency posture observed so far, covering *inventory coverage, detected cryptographic assets, and SBOM/CBOM-relevant records*. They form an **initial, still-evolving baseline** rather than a finished inventory.

4.1 Hypothesis Establishment

The working hypothesis adopted in this chapter is derived from **two complementary sources**: (i) the **technical discussions** conducted within the consortium and (ii) the **preliminary evidence** obtained from the inventorying activity. Together, these sources support a coherent rationale that develops in **three stages**. The first stage concerns the manner in which the examined environments communicate. The second, the location at which their cryptography is instantiated and the third, the requirements that their transition to quantum-resistant operation will impose:

1. The first stage is based on the observation that **the majority of the examined environments rely on TLS for network communication and authentication**. This was not assumed in advance, but was confirmed through the discussions with the use-case partners, in which TLS was consistently identified as the prevailing transport-security mechanism. This consensus establishes TLS as the primary focus of the inventory and, consequently, of the migration effort that follows from it.
2. The second stage addresses the **location at which the cryptography underlying TLS is instantiated**. The entry points identified during the inventorying are associated primarily with the cryptographic libraries present on each system, irrespective of whether a given library is invoked at

runtime. The presence of such a library is itself a relevant entry point, since cryptographic functionality is provided through several layers of the operating system and exposed through libraries that are integrated and consumed at the application level. Accordingly, the quantum exposure of a system is determined not by the operations observed at any single moment, but by the cryptographic capability that remains resident and reachable within it.

3. The third stage follows from the preceding two. The **legacy applications considered here will be required to migrate to a quantum-resistant authentication protocol**, namely PQ-TLS, and **this migration will in turn require crypto-agility across several distinct dimensions**. A detailed treatment of crypto-agility is provided in the subsequent sections at this point it is identified only as a necessary condition of the transition.

With respect to crypto-agility, two requirements may be distinguished. The first is the **accommodation of variable key sizes**. The second, and more general, is the **capacity to support and substitute different cryptographic primitives within the authenticated key exchange as the migration strategy evolves**. Within PiQASO, this capability is demonstrated in concrete terms: the implementation incorporates ML-KEM for key establishment and ML-DSA for signatures, so that the transition can be exercised against more than one post-quantum primitive.

These three propositions are not presented as assertions to be accepted without verification. Each is examined in Section 4.2, where it is assessed against the per-use-case inventory evidence and either validated or invalidated according to the findings obtained.

4.2 Results per Use Case

This section sets out the cryptographic asset inventory established for each PIQASO use case on the basis of the evidence furnished by the partners. The findings are reported use case by use case and are structured, in every instance, according to the CAI analysis modes defined previously—network analysis (NWA), software analysis (SFA), and package-manager analysis (PMA)—whereby the negotiated transport security, the cryptographic primitives embedded within the software, and the underlying library dependencies are each rendered explicit. Every tabular summary is accompanied by a concise interpretation that characterises the prevailing cryptographic posture and identifies the principal quantum-exposed elements, explaining why the mechanisms in use are susceptible to the quantum threat.

Table 4.2: UC1 (ABI) Sector: Automotive — cryptographic asset inventory

CAI Mode	Findings	Description	Recommendation
NWA	TLS_AES_256_GCM_SHA384	Transport is modern TLS 1.3 with AES-256-GCM so the symmetric layer is quantum-safe.	The cryptographic posture is therefore sound by classical standards but fully quantum-vulnerable at the key-exchange and signature layers, making this case a clear candidate for hybrid or post-quantum TLS and for the gradual substitution of RSA/ECDSA-based primitives through the PIQASO PQC SDK. The post-quantum functionalities selected for this use case are listed in Table 16.1.
SFA	ECDSA+SHA-256, RSA-OAEP+SHA-256, AES-GCM, RSA-2048, ECC P-256	The software-level evidence is comparatively rich, exposing a coherent classical stack built around ECDSA with SHA-256, RSA-OAEP with SHA-256, AES-GCM, RSA-2048, and the NIST P-256 curve.	

PMA	OpenSSL 3.5.4,libgcrypt20 1.11.0	The package layer is current, with OpenSSL 3.5.4 and libgcrypt20 1.11.0 installed.	
-----	----------------------------------	--	--

Table 4.3: UC2 (PAL) Sector: Automation — cryptographic asset inventory

CAI Mode	Findings	Description	Recommendation
NWA	TLS 1.3 (AES-256-GCM,ChaCha20-Poly1305)	The automation use case is captured from a single endpoint. Network analysis records TLS 1.3 with strong AEAD options (AES-256-GCM and ChaCha20-Poly1305)	Symmetric protection is robust, but the continued dependence on RSA-2048 for asymmetric operations marks the principal quantum-exposed element and the natural focus for post-quantum migration. The post-quantum functionalities selected for this use case are listed in Table 16.2.
SFA	AES-128 GCM/GMAC /CBC,RSA-2048	The software evidence shows AES in its GCM, GMAC, and CBC modes together with RSA-2048.	
PMA	OpenSSL 3.0.2,libgcrypt20 1.9.4	The package layer relies on OpenSSL 3.0.2 and libgcrypt20 1.9.4, both maintained but not the most recent releases.	

Table 4.4: UC3 (CXB) Sector: Finance: Secure Online Payments (e-wallet, e-banking) — cryptographic asset inventory

CAI Mode	Findings	Description	Recommendation
NWA	mTLS	exhibits mutually authenticated TLS (mTLS) at the network layer—a configuration that reflects the stringent endpoint-authentication requirements characteristic of the payments domain.	Although the available findings remain at a higher level of abstraction than those of the other cases, the reliance on mTLS necessarily implies classical certificate and key-exchange mechanisms beneath the transport layer, more granular SFA and PMA evidence would permit the specific asymmetric algorithms and certificate chains to be precisely characterised and prioritised for post-quantum readiness. The post-quantum functionalities selected for this use case are listed in Table 16.3.
SFA	SHA-2	The software evidence reports the SHA-2 family of hash functions	
PMA	OpenSSL 3.6.3	the package layer is maintained at a current OpenSSL 3.6.3 release.	

Table 4.5: UC4 (MOH-MORE-CERTH) Sector: Energy — cryptographic asset inventory

CAI Mode	Findings	Description	Recommendation
NWA	TLS 1.2/1.3; OPC UA Basic256Sha256	Network analysis observes TLS 1.2/1.3 alongside OPC UA using the Basic256Sha256 security policy.	This case therefore requires both a library-modernisation step and the introduction of quantum-resistant key exchange across its TLS and OPC UA channels. The post-quantum functionalities selected for this use case are listed in Table 16.4.

SFA	RSA-2048,ECDHE,ECDSA,ECC P-256,bcrypt+SHA-256	The software layer reveals a broad classical footprint, including RSA-2048, ECDHE, ECDSA, the ECC P-256 curve, and bcrypt with SHA-256 for credential protection.	
PMA	OpenSSL 1.1.1 / 3.x	The package layer is mixed, covering both OpenSSL 1.1.1 and 3.x; the presence of the end-of-life 1.1.1 branch is a notable migration constraint, since PQC-enabled libraries generally require the 3.x line.	

Table 4.6: UC5.1 (TELLU) Sector: Healthcare — cryptographic asset inventory

CAI Mode	Findings	Description	Recommendation
NWA	HTTP/1.1 (no TLS---unprotected transport)	The assessed endpoint communicates over plain HTTP/1.1, entirely without transport-layer security. This unprotected channel represents an immediate and tangible exposure, wholly independent of the prospective quantum threat.	The remediation priority is therefore first to enable TLS and thereby close the exposed transport channel and subsequently to migrate the classical asymmetric primitives toward hybrid or fully post-quantum equivalents. The post-quantum functionalities selected for this use case are listed in Table 16.5.
SFA	RSA,Ed25519,ECDSA,AES,SHA-256/512	The software layer, by contrast, proves cryptographically rich, encompassing RSA, Ed25519, ECDSA, ECDH over X25519, AES, and SHA-256/512.	
PMA	OpenSSL 3.0.2,libcrypt20 1.9.4	the package layer remains current with OpenSSL 3.0.2 and libcrypto20 1.9.4.	

Table 4.7: UC5.2 (PARTICLE) Sector: Healthcare — cryptographic asset inventory

CAI Mode	Findings	Description	Recommendation
NWA	TLS 1.3	Also within the healthcare domain, presents a compact and modern profile: TLS 1.3 at the network layer.	The configuration is clean and unencumbered by legacy-library constraints, so the migration path is correspondingly direct: principally, the replacement of ECDSA P-256 signatures with a post-quantum signature scheme such as ML-DSA, together with the adoption of hybrid key establishment within TLS. The post-quantum functionalities selected for this use case are listed in Table 16.5.
SFA	ECDSA P-256	ECDSA over the P-256 curve as the signature mechanism in software	
PMA	OpenSSL 3.0.13	Current OpenSSL 3.0.13 package base	

Table 4.8: UC6.1 (BIBA) Sector: Aerospace Integration — cryptographic asset inventory

CAI Mode	Findings	Description	Recommendation
NWA	None---plain MQTT, no TLS (unprotected transport)	The aerospace-integration use case yields a confirmed absence of cryptographic protection rather than missing data: network analysis observes plain MQTT with no TLS.	This case therefore demands attention on two fronts: the introduction of transport security over the MQTT channel as an immediate measure, and a subsequent design-level decision on where quantum-resistant protection should be embedded, given the current lack of any application cryptography. The post-quantum functionalities selected for this use case are listed in Table 16.6.
SFA	None detected	software analysis detects no cryptographic primitives.	
PMA	OpenSSL/libssl1.1 1.1.1w, libgcrypt 1.8.7	The only cryptographic material identified is at the package level, where the end-of-life OpenSSL 1.1.1w and libgcrypt 1.8.7 are installed but not exercised by the application.	

Table 4.9: UC6.2 (NEDHO) Sector: Aerospace Integration — cryptographic asset inventory

CAI Mode	Findings	Description	Recommendation
NWA	TLS_AES_256_GCM_SHA384	The second aerospace-integration case combines a modern transport layer—TLS 1.3 negotiated with TLS_AES_256_GCM_SHA384	The migration priority therefore centres on the asymmetric layer: the classical mechanism, RSA, is the principal quantum-exposed element and the natural candidate for replacement by a post-quantum signature scheme, complemented by hybrid key establishment within the already-modern TLS 1.3 channel. A confirmation of which of the detected primitives are actually exercised by the application would further refine the migration plan. The post-quantum functionalities selected for this use case are listed in Table 16.6.
SFA	RSA, SHA-256, HMAC, Ed25519	The software analysis surfaces a broad set of primitives, spanning contemporary mechanisms such as RSA, Ed25519, HMAC, and SHA-256 alongside a number of older algorithms exposed at the library level.	
PMA	OpenSSL 3.4.1, libgcr ypt20 1.11.0	current package base encountered in the inventory, namely OpenSSL 3.4.1 and libgcr ypt20 1.11.0.	

Table 4.10: UC7 (DPG) Sector: Online Media — cryptographic asset inventory

CAI Mode	Findings	Description	Recommendation
NWA	TLS_AES_256_GCM_SHA384	The network analysis identifies a modern TLS 1.3 transport layer secured by the TLS_AES_256_GCM_SHA384 cipher suite.	Classical mechanisms should be replaced with post-quantum signature schemes, alongside integrating hybrid key establishment into the existing TLS 1.3 network channel. Functionalities selected for this use case are listed in Table 16.7.

SFA	php-encryption, ext-openssl, ext-sodium, bcr ypt	The software analysis of the CMS exposes reliance on PHP's native cryptographic extensions (OpenSSL and libsodium) and specific application-level backend packages for encryption and password hashing.	Leverage of the PiQASO PQC SW SDK.
PMA	OpenSSL 3.5.5, libgcr ypt20 1.10.1-3	The package manager inventory establishes the baseline system cryptographic dependencies, explicitly identifying OpenSSL 3.5.5 alongside libgcr ypt20 1.10.1-3.	Leverage OpenSSL version higher than 3.5.6

Table 4.11: UC8 (ACCELI) Sector: UAVs — cryptographic asset inventory

CAI Mode	Practical Find-ings	Theoretic Find-ings	Description	Recommendation
NWA	TLS 1.2/1.3 — edge→backend, far edge→backend	far edge → edge runs over MAVLink	UAV use case spanning a resource-constrained edge node and a backend. The practical flows (edge ↔ backend, far-edge ↔ backend) run over TLS 1.2/1.3; the far-edge ↔ edge link runs over MAVLink. MAVLink1: Maximum Packet Size 263 bytes (6-byte header + up to 255-byte payload + 2-byte CRC); Minimum Packet Size 8 bytes (6-byte header + 0-byte payload + 2-byte CRC); Overhead 8 bytes per packet. MAVLink2: Maximum Packet Size 280 bytes (14-byte header + up to 255-byte payload + 2-byte CRC + up to 13-byte signature); Minimum Packet Size 12 bytes (10-byte header + 0-byte payload + 2-byte CRC); Overhead 14 bytes per packet.	The legacy libraries resident on the resource-constrained edge node thus represent the dominant constraint; and, since platforms of this class may be incapable of performing intensive post-quantum operations locally, the case emerges as a natural candidate for PQC-as-a-Service or hardware-accelerated offloading, in addition to a thorough modernisation of the underlying cryptographic libraries. The post-quantum functionalities selected for this use case are listed in Table 16.8.
SFA	RSA, SHA-256	not applicable in the first version	Software analysis shows classical RSA with SHA-256.	
PMA	libgcr ypt20 1.10.1; Jetson: Open SSL/libssl1.1 1.1 .1f, libgcr ypt20 1.8.5	not applicable in the first version	package base includes end-of-life OpenSSL 1.1.1f and older libgcr ypt	

Table 4.12: UC9 (STROWL) Sector: Transportation — cryptographic asset inventory

CAI Mode	Findings	Description	Recommendation
NWA	TLS_AES_256_GCM_SHA384	TLS 1.3 with TLS_AES_256_GCM_SHA384 at the network layer.	As with the other up-to-date cases, no legacy-library obstacle is present, and the migration effort accordingly centres upon two complementary measures: the introduction of hybrid or post-quantum key exchange within TLS, and the replacement of the ECDSA-based signatures with a quantum-resistant alternative. The post-quantum functionalities selected for this use case are listed in Table 16.9.
SFA	ECDSA	ECDSA as the software signature mechanism.	
PMA	OpenSSL 3.0.13, libcrypto20 1.10.3	package base comprising OpenSSL 3.0.13 and libcrypto20 1.10.3.	

Considered in aggregate, the per-use-case inventories set out in Tables 4.2–4.12 provide a first evidence-based characterization of the cryptographic posture in the PiQASO use cases. Hence, they permit the working hypothesis advanced in Section 4.1 to be weighed against the findings to which it gave rise. The predominant observation is uniform throughout. **The assessed environments rest almost entirely upon classical public-key cryptography.** Wherever network behaviour admitted of observation, the partners had, as a rule, adopted modern transport security. This consisted of **TLS 1.3**, coupled with authenticated-encryption suites, of which `TLS_AES_256_GCM_SHA384` recurs most frequently. However, the key establishment beneath it remained resolutely classical, predominantly *RSA-2048*, *ECDSA*, and *ECDHE over the NIST P-256 curve*. **No hybrid or post-quantum mechanism was returned by any captured flow.** The first proposition of Section 4.1 is accordingly upheld. That proposition held TLS to be the prevailing transport-security mechanism of the examined environments. A single qualification applies. *TLS is dominant without being exclusive*. OPC UA accompanies it in the energy use case, and a MAVLink link in the UAV case. Two environments afford **no transport-layer protection whatever**, the one conducted over plain HTTP/1.1, the other over unencrypted MQTT. Far from unsettling the majority findings, these exceptions delineate the precise bounds within which it obtains.

The software-level evidence reflects this same reliance upon classical primitives, principally *RSA*, *ECDSA*, *P-256*, *AES-GCM*, and *SHA-256*. It is at this layer that the second proposition of Section 4.1 receives its confirmation. That proposition held the quantum exposure of a system to be governed less by what it is observed to perform at any single instant than by **the cryptographic capability that remains resident and reachable within it**. The inventory bears the contention out without ambiguity. Package-manager analysis returned cryptographic libraries upon **every host examined**, irrespective of the activity discerned at the network or software layer. The aerospace-integration case furnishes the clearest illustration. Its network behaviour amounted to no more than plain MQTT, and its software analysis disclosed no cryptographic primitives. The host nonetheless retained **OpenSSL 1.1.1w and libcrypto 1.8.7**. In a limited number of cases the scans further exposed older algorithms at the library level. Whether these are actually invoked would warrant confirmation before any conclusion is drawn as to their operational relevance.

This final caveat may seem to sit uneasily beside the proposition just affirmed. The apparent tension dissolves once two distinct questions are held apart. The first is whether an asset belongs within the inventory and counts towards quantum exposure. This is settled by **presence alone**. A library that is resident and reachable enlarges the attack surface. It must therefore be entered into migration planning, whether or not it is exercised at the moment of observation. The second question concerns the order in which, and the urgency with which, the identified assets are subsequently to be treated. This is informed by **invocation**. Ascertaining which resident primitives lie upon active and operationally critical paths refines the sequence of remediation, without disturbing the inventory itself. Confirmation of invocation thus sharpens the migration plan rather than amending the catalogue of exposure. On this basis the inventory adopts *a precautionary posture for inclusion and an evidential one for prioritisation*.

The package-level perspective introduces a further constraint, and one of a decidedly concrete charac-

ter. With it comes the third proposition of Section 4.1. That proposition held that the surveyed applications must migrate toward a quantum-resistant authentication protocol, namely **PQ-TLS**, and that crypto-agility would be required of them in order to do so. Several partners already operate current releases of *OpenSSL 3.x* and recent versions of *libgcrypto*. Others remain bound to the **end-of-life OpenSSL 1.1.1 branch**, or to older *libgcrypto* releases. This circumstance directly curtails the feasibility of introducing PQC-enabled libraries in the absence of a preparatory upgrade. Three factors thus coincide: *a uniformly classical key-establishment layer, a heterogeneity of primitives and key sizes across the use cases, and these library constraints*. Together they establish at once the necessity of the migration and the agility its execution will demand. Within PiQASO that agility is rendered tangible rather than abstract. It is realised through the incorporation of **ML-KEM** for key establishment and **ML-DSA** for signatures, whereby the transition may be exercised across more than one post-quantum primitive. That this proposition rests, in the network-only captures, upon evidence yet maturing is openly acknowledged. Confirmation of the negotiated key-exchange group would place the absence of any post-quantum mechanism upon a fully evidential footing.

Certain cases prove additionally noteworthy on operational grounds, namely those services found to communicate **without any transport-layer protection**. One operates over plain HTTP/1.1, the other over unencrypted MQTT, the latter exhibiting no cryptographic primitives at the software level either. Such cases constitute *findings in their own right, and not mere lacunae in observation*. An unprotected transport represents **an immediate exposure, wholly independent of the prospective quantum threat**, and warrants remediation as a matter of priority. Taken collectively, the three propositions of Section 4.1 are upheld by the evidence. These results are most justly construed not as a finished inventory but as **an initial and expanding baseline**. They form the evidentiary foundation upon which the use-case-specific migration paths set out in the chapters that follow are ultimately founded.

Chapter 5

PiQASO UC1 – Sector: Automotive

5.1 Use case's Description, Goals and Objectives

Cybersecurity has become a critical concern for the automotive industry, which has undergone a profound digital transformation over the last decade. As in many traditional sectors of the economy, technological innovation is advancing at an unprecedented pace, driven by increasing connectivity, software-defined functionalities, cloud services, and autonomous driving technologies. Consequently, automotive cybersecurity is now one of the key enablers and drivers across the entire automotive value chain.

The continuous growth in computational capabilities, traditionally described by Moore's Law, together with the pervasive digitization of modern society, highlights the need to anticipate future cyber threats during the design phase of today's systems. Vehicles developed today are expected to remain operational for more than a decade, making it essential to address not only current cybersecurity risks but also those that may emerge in the future.

In this context, the ISO/SAE 21434:2021 standard, Road Vehicles – Cybersecurity Engineering, emphasizes the importance of identifying, assessing, and mitigating both known and foreseeable cyber threats throughout the vehicle lifecycle. The standard promotes a risk-based approach that supports the integration of cybersecurity activities into automotive engineering processes.

Looking ahead, the emergence of quantum computing introduces additional challenges for the automotive sector. Quantum computers may eventually be capable of breaking widely used public-key cryptographic algorithms, such as RSA and ECC, which currently protect vehicle communications, software updates, digital certificates, and backend infrastructures. This creates a "harvest now, decrypt later" risk, where encrypted data collected today could be decrypted in the future once sufficiently powerful quantum computers become available. As a result, automotive manufacturers and suppliers must begin evaluating the adoption of post-quantum cryptographic solutions to ensure the long-term security and resilience of connected and autonomous vehicles.

The PiQASO Automotive Use Case (UC1) investigates the adoption of Post-Quantum Cryptography (PQC) technologies to secure both V2X communications and the supporting Vehicular Public Key Infrastructure (vPKI). The use case focuses on Vehicle-to-Infrastructure (V2I) interactions involving vehicles, roadside infrastructure, backend services, and a managed vPKI operated by a specialised PKI Provider. Particular attention is devoted to authentication procedures, certificate lifecycle management, and secure key establishment mechanisms, which represent critical attack surfaces in the presence of future quantum-capable adversaries. The proposed approach evaluates how quantum-resistant cryptographic mechanisms can be integrated into existing ETSI-compliant V2X architectures while preserving interoperability, operational efficiency, and compatibility with current deployment models.

From a system perspective, the vehicle architecture considered in PiQASO is composed of multiple Elec-

tronic Control Units (ECUs) organised into functional zones and interconnected through in-vehicle communication networks (e.g., CAN-based architectures). The On-Board Unit (OBU) acts as the vehicle gateway towards external communication domains, including V2X networks, roadside infrastructure, and backend services. The scope of this use case is limited to securing communications involving the OBU and external entities. Internal ECU-to-ECU communications and safety-critical control functions are outside the scope of the PQC integration, as current automotive standards and deployment practices do not yet define the adoption of post-quantum cryptography for these internal vehicle functions. The focus is therefore placed on communications crossing the vehicle trust boundary, namely interactions between the OBU, roadside infrastructure, backend services, and the supporting vPKI ecosystem.

In modern C-ITS environments, vehicles equipped with OBUs continuously exchange safety-critical messages with infrastructure elements to enable advanced applications such as emergency vehicle priority management, intersection collision avoidance, and coordinated traffic flow optimization. These communications rely on the exchange of standardized V2X messages, primarily Cooperative Awareness Messages (CAM) and Decentralized Environmental Notification Messages (DENM). CAMs are transmitted periodically by vehicles to announce their presence, position, and kinematic status, while DENMs are event-triggered messages used to notify other road users of dangerous situations or special conditions.

The current security framework for V2X communications is based on a Vehicular Public Key Infrastructure (vPKI), which provides authentication, integrity protection, pseudonym management, and trust establishment through classical cryptographic mechanisms. Digital signatures are typically based on Elliptic Curve Cryptography (ECC), while secure communication channels rely on TLS-based authentication and classical key establishment mechanisms such as Elliptic Curve Diffie-Hellman (ECDH).

Although these mechanisms are considered secure against classical adversaries, they are vulnerable to future quantum attacks. In particular, Shor's algorithm would enable a sufficiently powerful quantum computer to compromise the mathematical assumptions underlying both RSA and ECC, threatening the long-term security of vehicular communication systems and the trust infrastructure that supports them.

The primary objective of this use case is to validate the adoption of post-quantum cryptographic mechanisms within automotive communication ecosystems and supporting vPKI infrastructures. To achieve this objective, PiQASO adopts a PQC-as-a-Service approach that abstracts cryptographic operations from vehicular applications while enabling the integration of NIST-standardised algorithms, including ML-DSA and SLH-DSA for digital signatures and ML-KEM for quantum-resistant key establishment.

The PiQASO approach adopts a service-oriented integration model that enables vehicular platforms to consume PQC-enabled cryptographic services and managed vPKI functionalities without requiring applications to directly interact with the underlying cryptographic implementations. This design philosophy addresses several critical challenges in the automotive domain:

- **Hardware Acceleration Requirements:** PQC algorithms are computationally more intensive than their classical counterparts. The identified architecture incorporates dedicated hardware acceleration via Field-Programmable Gate Arrays (FPGAs). The use case evaluates both software-only and hardware-accelerated execution paths in order to quantify the performance benefits achievable for post-quantum cryptographic operations.
- **Certificate Management Complexity:** The transition to PQC requires a complete overhaul of the certificate infrastructure. The use case explores both pure PQC and hybrid approaches, where classical and post-quantum cryptographic primitives coexist to ensure backward compatibility during the transition period.
- **Real-Time Performance Constraints:** Vehicular safety applications impose stringent timing requirements on communication and decision-making processes. While end-to-end application reactions are typically expected within a few hundred milliseconds, cryptographic operations must contribute

only a limited fraction of this budget. The PiQASO architecture therefore evaluates both software-based and hardware-accelerated PQC implementations to ensure that quantum-resistant security can be achieved without compromising operational requirements.

The reference implementation considered in PiQASO deploys the PiQASO SDK on Linux-based OBU and RSU platforms. The SDK enables automotive and roadside applications to consume PQC-enabled services and, where applicable, to interface with local hardware acceleration resources.

The use case considers two representative potential application scenarios that demonstrate the practical value of quantum-resistant V2I communications:

1. **Emergency Vehicle Priority Management:** An ambulance or other emergency vehicle broadcasts high-priority CAM messages to the RSUs along its route. Upon receiving and authenticating these messages, the RSUs automatically adjust traffic light timing to create a priority lane for the emergency vehicle. The integrity and authenticity of these messages are critical, as false emergency signals could lead to traffic disruptions or accidents.
2. **Intersection Safety Coordination:** RSUs coordinate vehicle flow at intersections by exchanging DENM messages with approaching vehicles. These messages contain time-critical information regarding right-of-way, potential collision zones, and recommended speeds. The security of these messages is paramount to prevent malicious actors from causing accidents or traffic chaos.

Beyond demonstrating the feasibility of PQC-enabled V2I communications, the use case aims to evaluate the operational, performance, interoperability, and migration aspects required for the adoption of quantum-resistant security mechanisms within future automotive ecosystems. The use case aims to:

- Validate the feasibility of integrating PQC into resource-constrained automotive platforms while meeting real-time performance requirements.
- Demonstrate the practical implementation of a managed vPKI infrastructure that supports the issuance, distribution, and revocation of PQC certificates.
- Establish benchmarks for PQC operation latencies in vehicular contexts, including signature generation, signature verification, and key establishment.
- Evaluate the impact of increased message sizes due to PQC signatures on wireless channel utilization and message dissemination performance.
- Support the development of a migration strategy that allows for a gradual transition from classical to quantum-resistant cryptography without disrupting existing vehicular communication systems.
- Assess interoperability between different PQC software stacks, cryptographic libraries, and managed vPKI infrastructures implementing NIST-standardised algorithms.
- Provide an architectural foundation that can be extended to quantum-secure backend communication and software update workflows based on the same PQC trust infrastructure.

5.2 PiQASO Functionalities

Table 5.1: PiQASO Functionalities to be evaluated for Use Case 1 (Automotive)

PiQASO Functionality	Description and Purpose in the Use Case	RP1	RP2
PQ Signatures (ML-DSA, FALCON)	Used by the OBU and RSU to generate and verify signature tokens for mutual authentication, and to sign CAM/DENM messages, ensuring authenticity and integrity of V2X communications.	✓	
PQ-TLS (incorporating ML-KEM)	Establishes a quantum-resistant secure channel between the OBU and the managed vPKI for certificate provisioning, and between OBU and RSU for authenticated V2I sessions and shared-secret establishment.	✓	
PQCaaS	Provides the managed vPKI and OTA Update Server with PQC-as-a-Service capabilities, abstracting certificate issuance, revocation, and cryptographic operations from vehicular and road-side applications.	✓	
QR-TCB	Secure storage of cryptographic keys and supporting the execution of PiQASO PQC enablers.	✓	✓
Hardware Acceleration	Offloading of ML-DSA and ML-KEM operations to the FPGA-based accelerator on the OBU/RSU, benchmarked against a software-only execution path to meet V2X real-time timing constraints.		✓
AKE	Authenticated Key Exchange combining ML-DSA-based mutual authentication with ML-KEM-based key encapsulation to derive session keys for protected unicast OBU↔RSU communications.		✓
PQ Signatures (Alternative primitives: FALCON/FN-DSA)	Evaluation of alternative post-quantum signature algorithms for OTA firmware metadata and package signing, assessing signature size and verification latency versus the ML-DSA baseline.		✓

5.3 Managed PQC-enabled vPKI Infrastructure

The security of both current and future V2X ecosystems relies on the availability of a trusted Vehicular Public Key Infrastructure (vPKI) capable of managing identities, certificates, trust relationships, and revocation information for all participating entities. Within the PiQASO Automotive Use Case, the vPKI constitutes the trust foundation supporting both classical and post-quantum communication workflows.

The managed vPKI infrastructure considered in this use case is provided through Aruba's Registration Authority (RA) and Certification Authority (CA) services. The infrastructure supports the issuance and management of certificates based on both classical and post-quantum cryptographic algorithms through a common operational framework. This approach enables the evaluation of PQC-enabled certificate lifecycle management while maintaining compatibility with existing PKI deployment models and migration strategies.

The vPKI infrastructure is responsible for authenticating requesting entities, validating certificate requests, issuing certificates, managing certificate status information, and maintaining the trust relationships required by vehicles, roadside infrastructure, backend services, and software update systems. As a result, the same trust infrastructure can support both V2X communications and additional services such as secure OTA software update workflows.

5.3.1 vPKI Architecture and Actors

The managed vPKI infrastructure is composed of four main functional entities: the Client, the Registration Authority (RA), the Certification Authority (CA), and the Certificate and Audit Store.

The Client represents the requesting entity, which may correspond to an On-Board Unit (OBU), a Road-side Unit (RSU), a backend component, or any other authorised participant in the automotive ecosystem. The client is responsible for generating the cryptographic key pair locally, creating the Certificate Signing Request (CSR), storing the issued certificate, and subsequently using the certificate within operational workflows.

The Registration Authority (RA) acts as the primary interface between clients and the trust infrastructure. The RA authenticates and authorises requests, validates certificate information, enforces certificate policies, records audit information, and coordinates certificate issuance procedures.

The Certification Authority (CA) is responsible for issuing and signing certificates according to the selected certificate profile and cryptographic algorithm. Separate issuing authorities may be used depending on the certificate type and algorithm family. The Certificate and Audit Store maintains information regarding issued certificates, certificate status, lifecycle events, and audit logs required for operational traceability and trust management.

A key design principle of the Aruba-managed vPKI is the adoption of request-based enrolment, whereby private keys are generated and retained exclusively by the client. The trust infrastructure never receives or stores client private keys, thereby preserving cryptographic ownership and reducing key compromise risks.

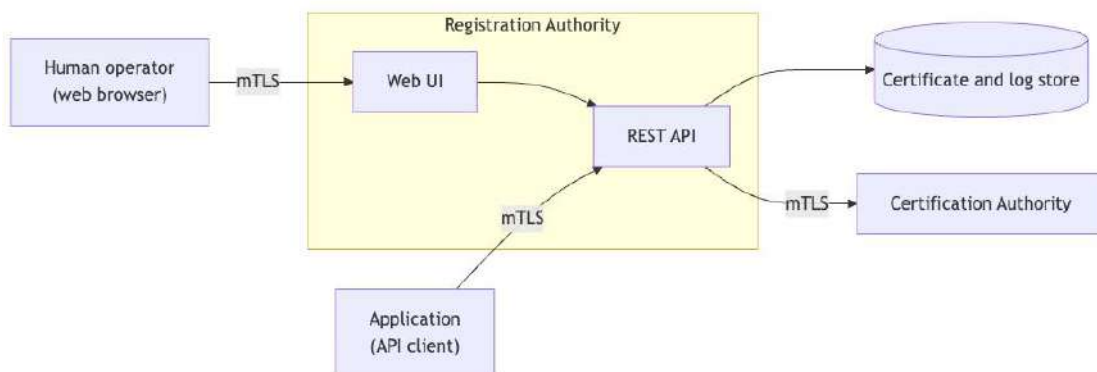


Figure 5.1: Automotive vPKI Architecture

5.3.2 Certificate Issuance Workflow

Certificate issuance follows a request-based workflow involving the client, the Registration Authority, and the Certification Authority. The same operational procedure is applied to both classical and post-quantum certificates, with the selected algorithm family acting as a request parameter.

The process begins with the client generating a local cryptographic key pair. The private key remains under the exclusive control of the client, while the corresponding public key is included within a Certificate Signing Request (CSR). The client subsequently submits the CSR to the Registration Authority through mutually authenticated communication channels.

Upon receiving the request, the Registration Authority authenticates the client, validates the request contents, verifies compliance with the applicable certificate profile, and authorises certificate issuance.

Following successful validation, the request is forwarded to the Certification Authority responsible for the selected certificate type and algorithm family.

The Certification Authority generates and signs the requested certificate before returning it to the Registration Authority. The issued certificate, together with the corresponding certificate chain and issuance information, is then returned to the client and recorded within the audit infrastructure.

The architecture supports multiple algorithm families, including classical RSA and ECC certificates as well as post-quantum ML-DSA and SLH-DSA certificates. This flexibility enables the evaluation of different migration strategies and interoperability scenarios within the PiQASO Automotive Use Case.

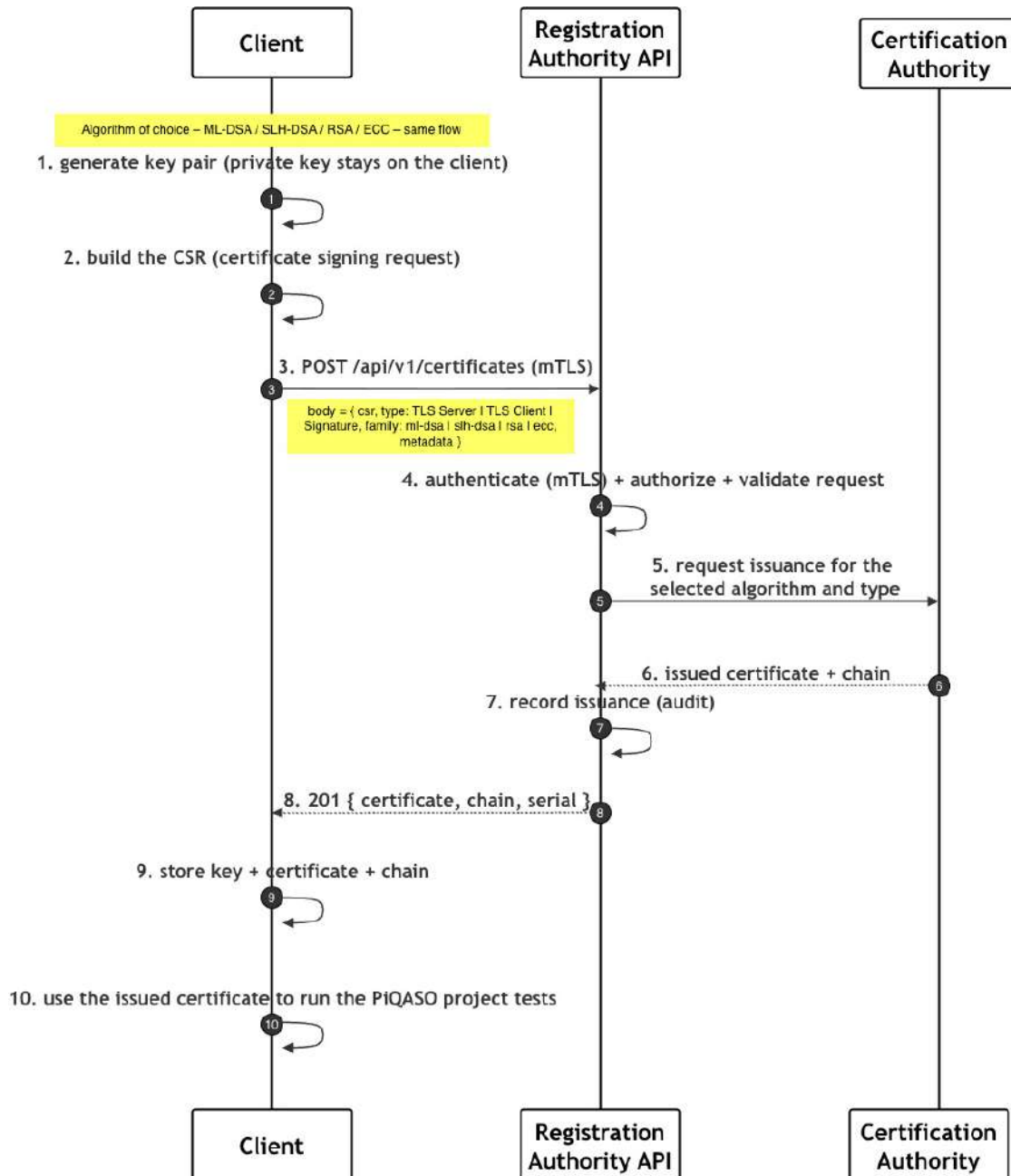


Figure 5.2: Automotive vPKI Sequence Diagram

Figure 5.2 illustrates the certificate issuance workflow implemented by the managed vPKI infrastructure.

5.3.3 Certificate Lifecycle Management

Beyond certificate issuance, the managed vPKI infrastructure supports the complete lifecycle management of certificates throughout their operational validity period.

Supported lifecycle operations include certificate issuance, renewal, revocation, deletion, and certificate status queries. These operations are exposed through dedicated Registration Authority interfaces and follow the same authentication and authorisation principles adopted during certificate enrolment.

Certificate renewal enables entities to obtain updated credentials before certificate expiration while preserving operational continuity. Certificate revocation provides mechanisms for invalidating compromised, expired, or otherwise untrusted credentials. Certificate status information is subsequently made available to relying parties to support trust validation decisions during communication establishment and message verification procedures.

The availability of lifecycle management services is essential for maintaining trust within large-scale V2X ecosystems, where vehicles, roadside infrastructure, backend services, and software update platforms continuously depend on accurate and up-to-date certificate status information.

The vPKI capabilities described in this section constitute the trust foundation upon which both the AS-IS and TO-BE automotive communication architectures presented in the following sections are built.

5.4 "As-Is" Scenario

The current state of vehicular communication security is based on established cryptographic standards and protocols. Understanding this baseline is essential to appreciate the challenges and requirements driving the transition to post-quantum cryptography.

5.4.1 System Model and Current Security Architecture

The current automotive communication ecosystem is composed of multiple interconnected domains spanning the vehicle, roadside infrastructure, backend services, and the Vehicular Public Key Infrastructure (vPKI). Within the vehicle, several Electronic Control Units (ECUs) are organised into functional zones and interconnected through in-vehicle communication networks such as CAN or Automotive Ethernet. These components support vehicle sensing, control, diagnostics, and application-specific functionalities.

The On-Board Unit (OBU) acts as the communication gateway between the vehicle and external entities. Through the OBU, vehicles interact with Roadside Units (RSUs), backend services, and trust infrastructure components. While internal ECU-to-ECU communications are essential for vehicle operation, they are outside the scope of this use case, which focuses on communications crossing the vehicle trust boundary.

The main components involved in the AS-IS scenario are:

- Vehicle ECUs and Zonal Controllers: responsible for vehicle control functions, sensor processing, and internal communications.
- In-Vehicle Network: communication infrastructure interconnecting ECUs, zonal controllers, and the OBU.
- On-Board Unit (OBU): communication gateway responsible for external connectivity and V2X communications.

- Roadside Unit (RSU): roadside infrastructure component supporting traffic management and V2I services.
- Backend Services: external systems supporting traffic monitoring, operational management, and service provisioning. Vehicular Public Key Infrastructure (vPKI): trust infrastructure responsible for certificate issuance, validation, pseudonym management, and certificate revocation services.

The security architecture currently adopted in V2X ecosystems relies on classical public-key cryptography and ETSI-compliant vPKI services. Authentication and integrity protection are provided through pseudonym certificates and Elliptic Curve Digital Signature Algorithm (ECDSA) signatures, while secure communication channels rely on TLS-based authentication and classical key establishment mechanisms. The vPKI manages the lifecycle of enrolment and pseudonym certificates, enabling trusted participation in the V2X ecosystem while preserving vehicle privacy.

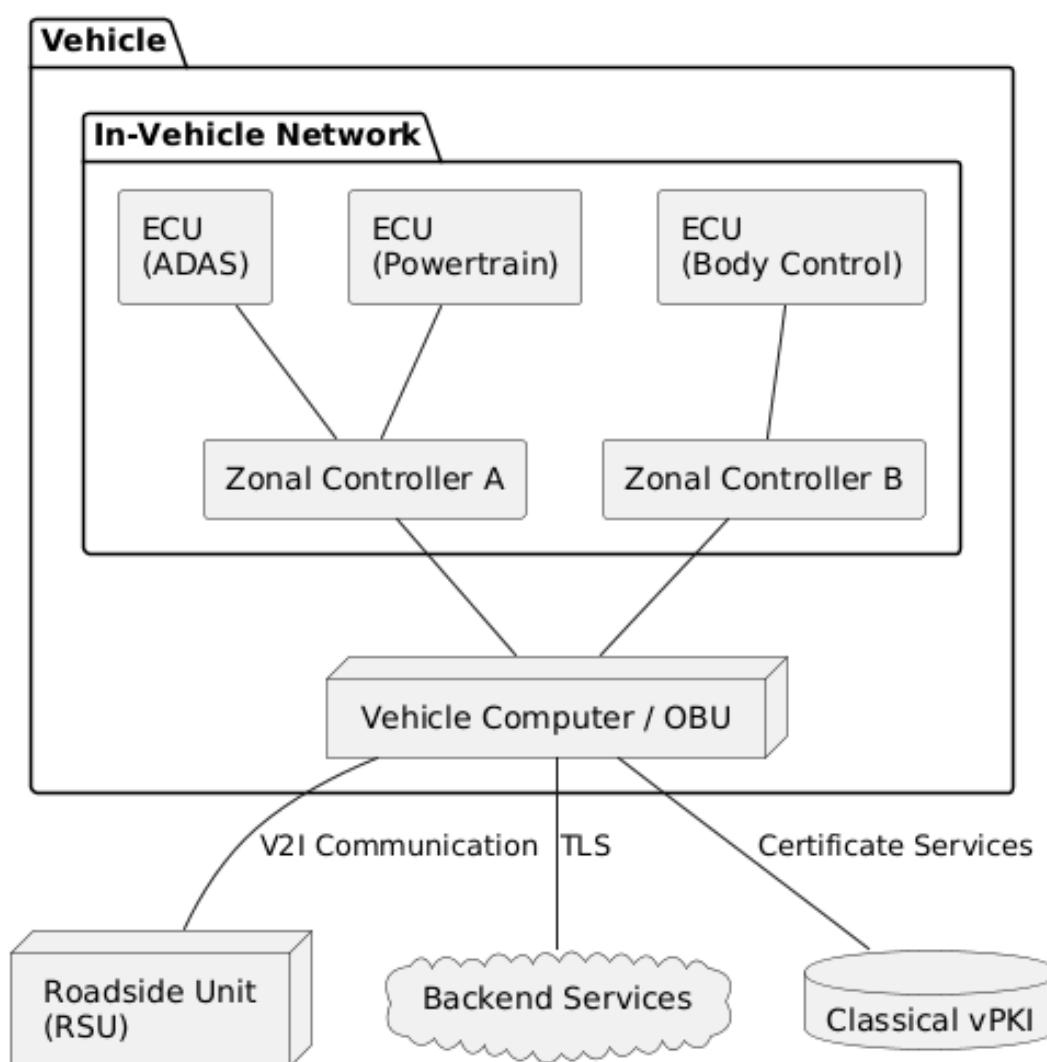


Figure 5.3: Automotive AS-IS System Architecture

Figure 5.3 illustrates the system architecture characterising the current V2X ecosystem. For readability purposes, sensors and actuators are abstracted within the corresponding ECUs, while the OBU represents the vehicle gateway towards external communication domains, including roadside infrastructure, backend services, and the supporting vPKI. Trust establishment relies on a classical vPKI infrastructure that provides certificate lifecycle management services for vehicles and roadside infrastructure. The

following paragraphs describe the main security principles and operational mechanisms adopted in the AS-IS scenario.

The security architecture operates on several fundamental principles:

- **Certificate-based Authentication:** Every vehicle and infrastructure element participating in V2X communications must enroll in the vPKI. During the enrollment process, the vehicle or RSU is issued a long-term enrollment certificate that serves as a permanent identity within the system. This certificate, however, is never used directly for signing V2X messages to protect privacy.
- **Pseudonymity for Privacy Preservation:** To prevent vehicle tracking through their transmissions, the vPKI issues pools of short-term pseudonym certificates to each enrolled entity. These certificates are cryptographically unlinkable to each other and to the long-term enrollment certificate, except by the vPKI itself. Vehicles rotate through these pseudonym certificates during operation, typically changing certificates every few minutes or when crossing significant geographic boundaries.
- **Classical Cryptographic Primitives:** The current implementation uses ECDSA (Elliptic Curve Digital Signature Algorithm) with the NIST P-256 curve for all digital signatures. This choice provides a good balance between security (approximately equivalent to 128-bit symmetric security) and computational efficiency. Public key operations can be performed efficiently even on modest embedded processors.

”As-Is” vPKI Architecture The AS-IS system model, as depicted in Figure 5.4, represents the VPKI as a set of logical trust services supporting the authentication of C-ITS stations in the automotive scenario. The OBU and RSU are modelled as the main stations interacting with the VPKI. The Registration Authority supports the registration of stations by the station operator, while the Enrolment Authority issues enrolment certificates used to identify legitimate stations within the system. The Authorisation Authority provides pseudonym certificates that are then used by the OBU and RSU to sign V2X messages. The Root CA and related trust services are represented at an aggregated level, as their role is to provide the trust anchor (i.e., supporting the secure management of the short-lived issued credentials) for the VPKI hierarchy. This simplified view keeps the focus on the entities that are relevant for the AS-IS automotive use case, without introducing implementation-specific details.

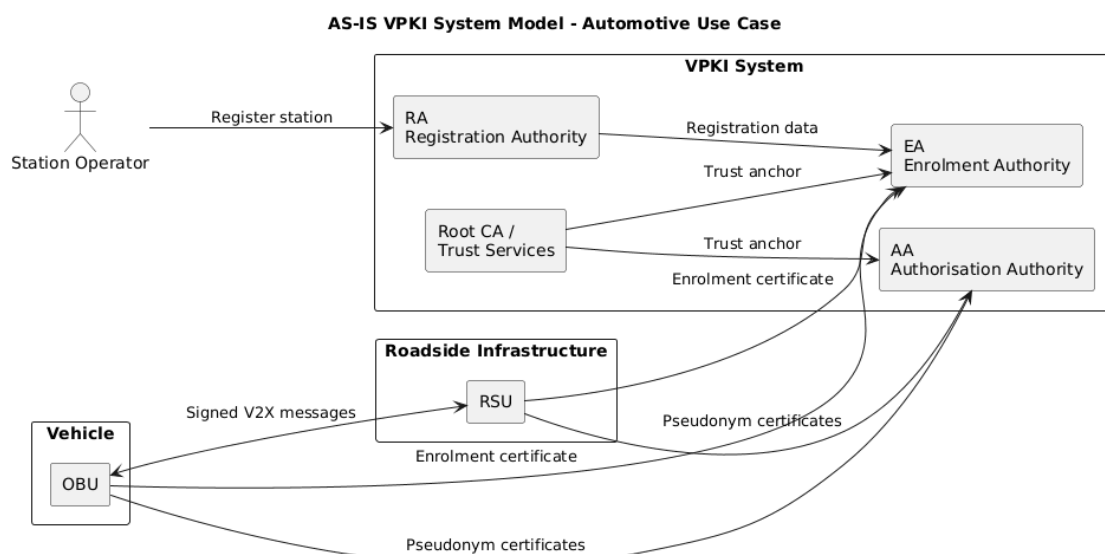


Figure 5.4: Automotive AS-IS vPKI Architecture

5.4.2 Communication Workflow

The operational flow in the AS-IS scenario proceeds through several distinct phases, as illustrated in the AS-IS sequence diagram. The diagram focuses on the interactions among vehicles, roadside infrastructure, and the classical vPKI from a system-level perspective. Internal certificate management procedures performed by the vPKI are intentionally abstracted and represented as a trust service. A detailed description of the certificate lifecycle management process is provided separately through the vPKI-specific workflow supplied by the PKI Provider.

Phase 1: Certificate Provisioning and Trust Establishment. Before participating in V2X communications, both OBUs and RSUs must obtain valid certificates issued by the vPKI. Periodically, vehicles and infrastructure elements request new pseudonym certificates and updated revocation information. The vPKI validates the requesting entity, verifies its enrolment status, and provides the required certificate material together with certificate status information. The vPKI is responsible for managing the complete certificate lifecycle, including certificate issuance, validation, renewal, and revocation processes. These functions provide the trust foundation upon which V2X communications rely.

The output of this process is a set of valid certificates that can subsequently be used for authentication and secure communications within the V2X ecosystem.

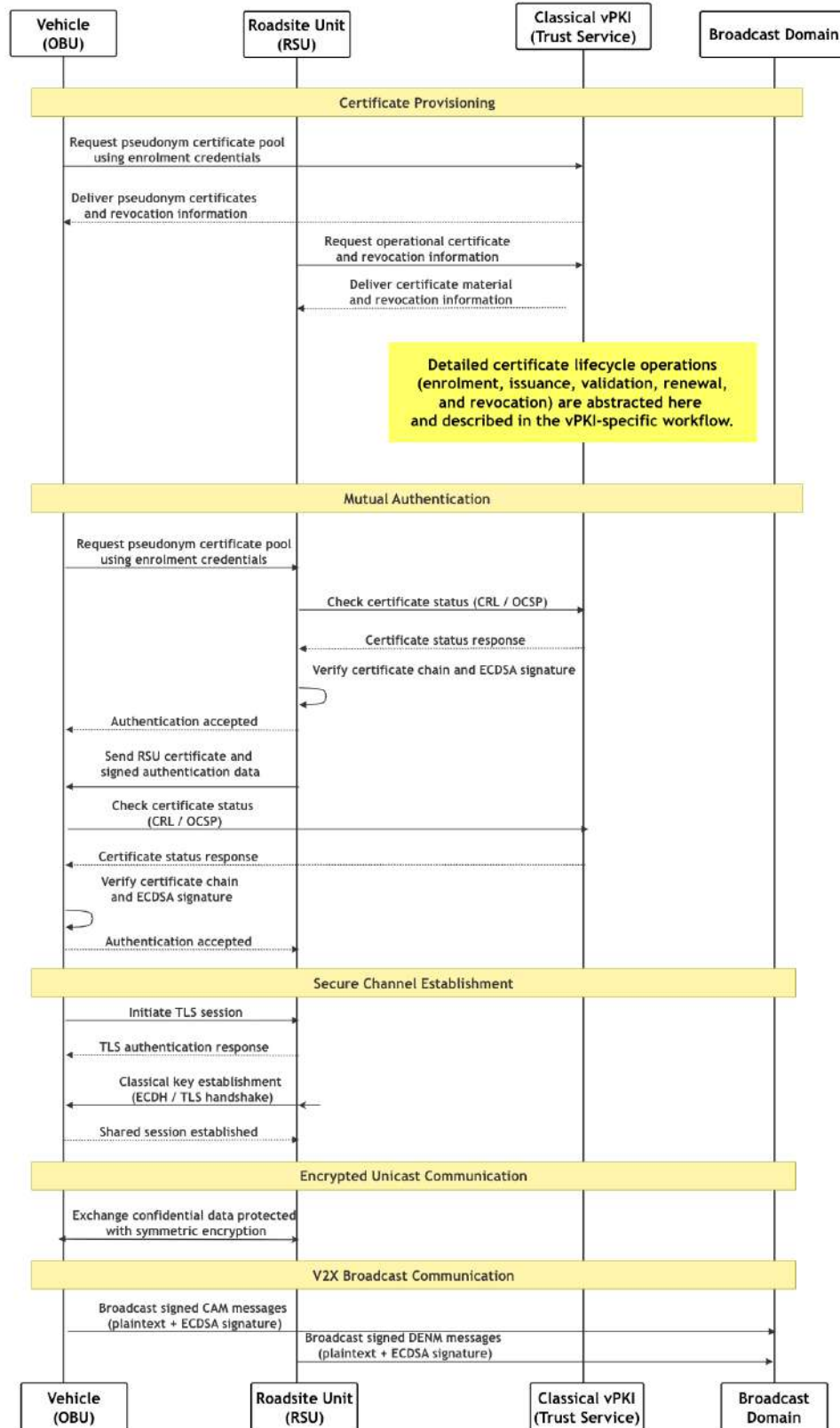


Figure 5.5: Automotive AS-IS Sequence Diagram

Phase 2: Certificate Exchange and Mutual Authentication. When an OBU initiates communication with an RSU, both entities perform mutual authentication using their respective certificates. The OBU first transmits its current pseudonym certificate together with signed authentication data. Upon reception, the RSU performs a series of verification steps:

- Validation of the certificate structure and certification chain.
- Verification of the certificate validity period.
- Verification of the certificate revocation status through CRL or OCSP mechanisms.
- Verification of the digital signature using the public key contained in the certificate.

If all checks are successful, the OBU is considered a trusted participant. The RSU then performs the same procedure towards the OBU by presenting its own certificate and authentication data. Once both parties have successfully completed the verification process, a trusted communication relationship is established.

Phase 3: Secure Channel Establishment. For communications requiring confidentiality, the OBU and RSU establish a secure communication channel using TLS-based authentication and classical key establishment mechanisms. Depending on the deployment architecture, this process may rely on Elliptic Curve Diffie-Hellman (ECDH) or equivalent classical approaches supported by the communication stack.

Upon successful completion of the handshake procedure, both parties derive a shared symmetric session key. This key is subsequently used to protect confidential communications exchanged between the vehicle and roadside infrastructure.

Phase 4: Encrypted Unicast Communication. Once a secure channel has been established, the OBU and RSU can exchange confidential information such as diagnostic data, configuration parameters, operational logs, or service-specific information. These communications are protected using symmetric cryptographic algorithms such as AES operating in authenticated encryption modes.

This approach guarantees confidentiality while also protecting message integrity and authenticity against modification attempts performed by external adversaries.

Phase 5: V2X Broadcast Communication. The final phase concerns the dissemination of safety-related V2X messages. Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs) are broadcast in plaintext to ensure that they can be received by all nearby vehicles and infrastructure elements without requiring prior establishment of shared secrets.

To guarantee authenticity and integrity, each message is digitally signed using the private key associated with the sender's current pseudonym certificate. The transmitted packet therefore contains the message payload, the digital signature, and the corresponding certificate information.

Receiving entities validate the certificate status and verify the signature before accepting the message as trustworthy. Messages associated with invalid signatures or revoked certificates are discarded.

5.4.3 As-Is vPKI Certificate Provisioning Workflow

The sequence diagram provided in Figure 5.6 provides a focused view of the AS-IS VPKI certificate provisioning process for the automotive use case. Instead of representing the VPKI as a single logical entity, the flow distinguishes the Registration Authority, Enrolment Authority, and Authorisation Authority. The station operator first registers the OBU and RSU through the RA, which provides the relevant registration information to the EA. The registration and enrolment steps are intended as preliminary or periodic lifecycle operations, typically performed during station rollout, provisioning, or certificate renewal, and should not be interpreted as actions triggered by an operator every time the OBU or RSU interacts with the VPKI or participates in V2X communication. The output of these steps is a registration token binded to the long-term identity of the OBU and the RSU, and is circulated back to the intended holder (the OBU/RSU). The OBU and RSU then request enrolment certificates from the CA (presenting the issued registration token signed by the registration authority). The OBU and RSU then request enrolment certificates from the

EA, allowing them to be recognised as legitimate C-ITS stations. Based on these enrolment credentials, each station requests a pool of short-term pseudonym certificates from the AA, together with the relevant revocation information. These certificates are then used in the broader AS-IS communication flow to sign V2X messages exchanged between OBU and RSU.

The endmost goal of this operational workflow, capturing the internal specifics of the vPKI architecture, is to also highlight that migration to the PQC does not only entail equipping the CCAM entities (i.e., OBU and RSU) with the necessary PQC SDK primitives, but also elevating the backend infrastructure with the respective PQC capabilities. This is also going to be tested in the context of PiQASO, based on the onboarding of a testing vPKI infrastructure, leveraging the well-established **EJBCA vPKI**.

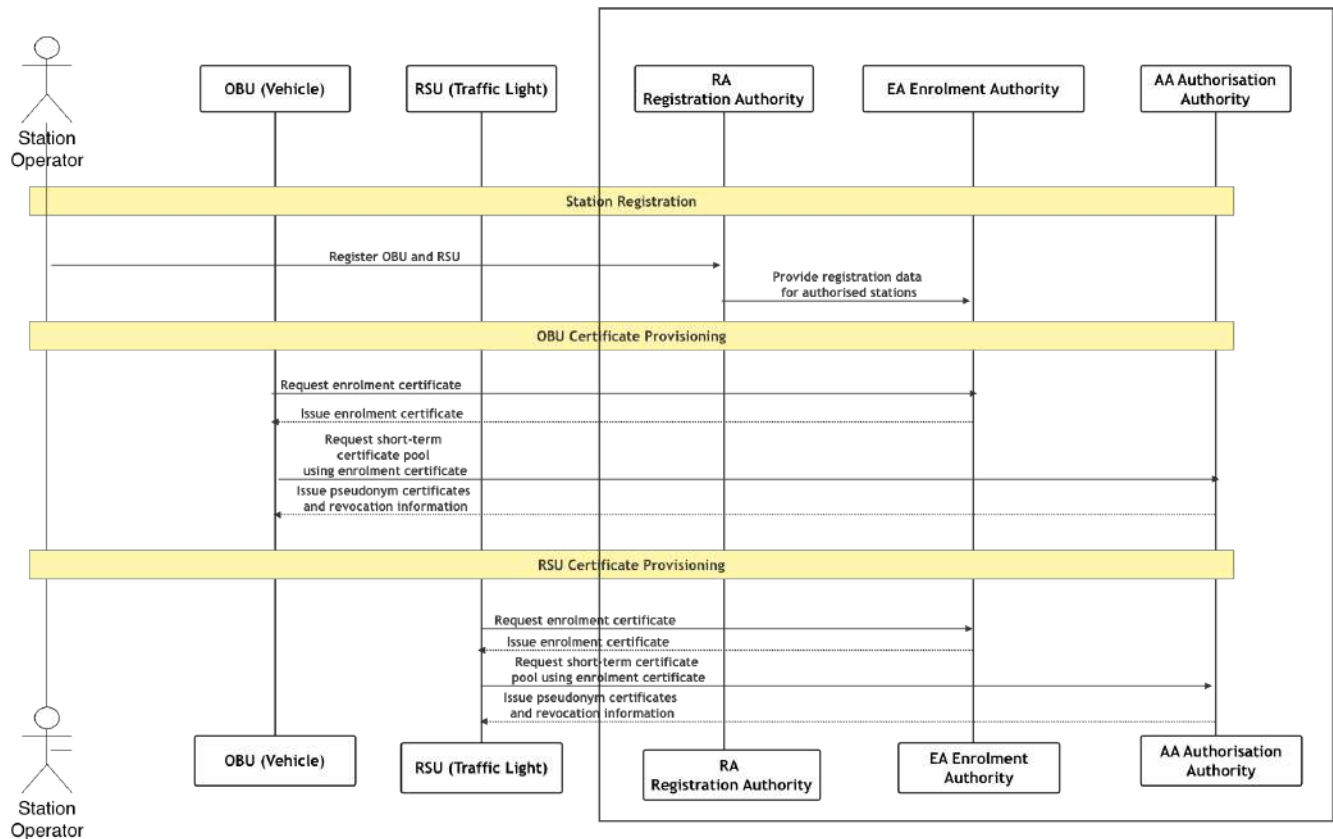


Figure 5.6: Automotive AS-IS vPKI Certificate Provisioning Sequence Diagram

5.4.4 Limitations of the Current Security Architecture

The security architecture described in the AS-IS scenario reflects the current state of deployment of V2X ecosystems and is largely aligned with existing ETSI standards and operational practices. While this architecture provides an effective level of protection against classical adversaries, several limitations emerge when considering the long-term impact of quantum computing on public-key cryptography.

The first limitation concerns the dependence on classical asymmetric cryptographic algorithms. Authentication mechanisms, certificate infrastructures, and secure channel establishment procedures rely on cryptographic primitives whose security is based on mathematical problems that could be efficiently solved by sufficiently powerful quantum computers. As a consequence, the trust assumptions underpinning the current vPKI architecture may become invalid in the future.

A second limitation is related to the long operational lifetime of automotive systems. Vehicles often remain in service for more than ten years, while trust infrastructures and certificate management systems may

need to remain valid for even longer periods. This creates a potential exposure to “harvest now, decrypt later” scenarios, in which communications or cryptographic material collected today could be exploited once practical quantum computing capabilities become available.

The current architecture also relies entirely on classical certificate lifecycle management procedures. Although effective under current threat models, these mechanisms do not provide quantum-resistant guarantees for certificate provisioning, authentication, or trust establishment processes. Consequently, the security of the overall ecosystem remains dependent on cryptographic assumptions that may not remain valid over the vehicle lifecycle.

From an interoperability perspective, the existing architecture does not provide a structured migration path towards quantum-resistant cryptography. Introducing PQC algorithms into operational V2X ecosystems requires maintaining compatibility with existing standards, communication protocols, and deployment models while minimizing disruption to currently deployed infrastructures.

Finally, the adoption of PQC introduces additional computational and communication overheads that are not addressed by current architectures. Larger key sizes, larger signatures, and increased computational complexity require careful evaluation of performance impacts on automotive platforms and roadside infrastructure components.

These limitations motivate the transition towards the TO-BE architecture proposed in the PiQASO Automotive Use Case, where post-quantum cryptographic mechanisms, managed PQC-enabled vPKI services, and hardware-assisted cryptographic acceleration are introduced to support the evolution towards quantum-resistant V2X ecosystems.

5.5 “To-Be” Scenario: PiQASO Architecture for Post-Quantum Security

The limitations identified in the AS-IS architecture motivate the adoption of a quantum-resistant security framework capable of preserving the long-term trustworthiness of V2X ecosystems. The TO-BE scenario proposed within PiQASO introduces Post-Quantum Cryptography (PQC) technologies, managed PQC-enabled trust infrastructures, and hardware-assisted cryptographic acceleration to support the transition towards quantum-secure automotive communications.

The proposed architecture maintains compatibility with existing V2X communication models while introducing quantum-resistant mechanisms for authentication, certificate lifecycle management, and secure key establishment. In addition, the architecture leverages the PiQASO SDK and PQC-as-a-Service framework as an integration layer between automotive applications, roadside infrastructure, cryptographic services, and managed vPKI platforms.

The objective of the TO-BE architecture is not to modify the internal operation of vehicle control systems but rather to secure communications crossing the vehicle trust boundary. Consequently, existing in-vehicle communications among ECUs and safety-critical functions remain unchanged, while PQC technologies are introduced at the level of the OBU, roadside infrastructure, backend services, and supporting trust infrastructures.

The following sections describe the target system architecture, the enhanced communication workflow, and the expected benefits introduced by the adoption of PiQASO technologies within the automotive ecosystem.

5.5.1 System Model and Target Architecture

The TO-BE architecture extends the AS-IS V2X system model by introducing PiQASO components for post-quantum cryptographic services, hardware-assisted acceleration, and integration with a managed PQC-enabled vPKI. The target architecture preserves the separation between the internal vehicle domain and the external communication domains, while strengthening the security of all interactions crossing the vehicle trust boundary.

Within the vehicle, ECUs and zonal controllers remain responsible for sensing, actuation, diagnostics, and control-related functions. These components continue to communicate through existing in-vehicle networks such as CAN or Automotive Ethernet. The integration of PQC technologies does not modify the internal ECU-to-ECU communication model or the operation of safety-critical vehicle functions. Instead, PQC mechanisms are introduced at the level of the OBU, which acts as the gateway between the vehicle and external entities.

The OBU hosts the PiQASO SDK, which acts as the integration layer between automotive applications, external PQC services, and the supporting trust infrastructure. Through this layer, applications can request cryptographic operations such as signature generation, signature verification, certificate handling, and secure key establishment without directly managing the implementation details of the selected PQC algorithms. The SDK interfaces both with local hardware acceleration resources and with the external PiQASO PQC-as-a-Service components.

To meet the timing constraints of V2X communications, the OBU interfaces with a dedicated FPGA-based accelerator. The accelerator supports computationally intensive PQC operations, including ML-DSA signature generation and verification and ML-KEM encapsulation and decapsulation. A software-only execution path may also be maintained to support benchmarking, fallback operation, and comparison between accelerated and non-accelerated implementations.

The RSU follows the same integration model adopted within the vehicle domain. Roadside communication functions interface with the PiQASO SDK, which provides access to PQC-enabled cryptographic services and interacts with local FPGA-based acceleration resources when required. This enables roadside infrastructure to support quantum-resistant authentication, certificate validation, secure key establishment, and signed dissemination of V2X messages while maintaining compatibility with the timing constraints of traffic management applications.

The managed PQC-enabled vPKI provides the trust infrastructure for the target architecture. It supports certificate issuance, validation, renewal, and revocation services for both vehicles and roadside infrastructure while enabling the gradual transition from classical to post-quantum trust models. The OBU and RSU interact directly with the managed vPKI for certificate lifecycle operations, while PiQASO components provide the integration layer for PQC-enabled cryptographic services and application-level adoption.

Backend services remain part of the architecture and support traffic management, operational monitoring, service provisioning, and potential software update workflows. Secure communications with backend services and trust infrastructures may rely on PQC-enabled or hybrid secure channel mechanisms, depending on the maturity of the selected protocols and the deployment configuration.

The target architecture adopts a distributed key-management model spanning backend services, OTA infrastructure, OBUs, and in-vehicle components. Cryptographic material is distributed across three main domains: backend systems, vehicle platforms, and internal vehicle components. Backend services manage service-level credentials and the integration key used to establish the trust relationship with the OTA infrastructure. The OBU manages certificates, authentication credentials, and the cryptographic material required for secure external communications, while ECU-specific keys remain confined to the internal vehicle domain.

For OTA software update workflows, an integration key links the backend environment with the OTA Update Server. The OTA process follows a two-step approach. First, a trusted baseline is established through PQC-enabled authentication and certificate validation. Subsequently, the OBU interacts with the PiQASO framework to perform ML-KEM-based key establishment, enabling the creation of a protected communication context for update delivery. Rather than focusing on the management of individual keys, the validation activities evaluate the overall flow OBU → PQCaaS → ML-KEM and the delivery of software updates over a PQC-authenticated channel.

Following successful authentication and key establishment, a separate symmetric key is derived and used by the OBU to protect and verify OTA update operations, including firmware metadata validation and firmware package verification. ECU keys are used exclusively for internal vehicle communications and are not involved in external V2I, vPKI, or OTA communication workflows.

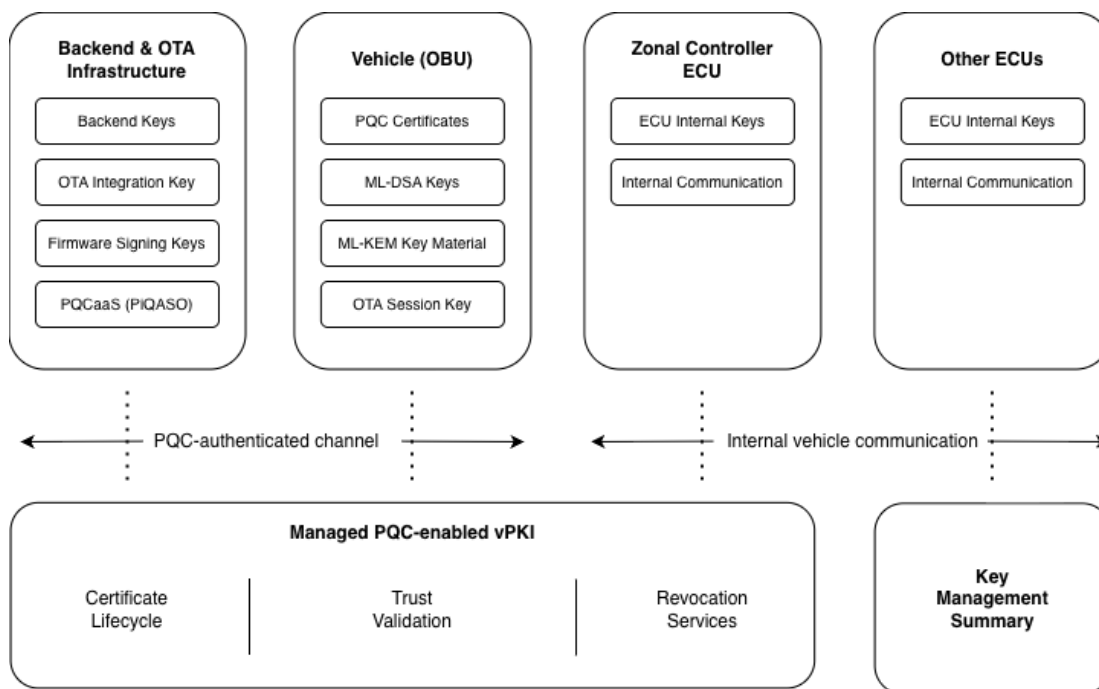


Figure 5.7: Automotive TO-BE Key Management

Figure 5.7 illustrates the key-management model adopted within the PiQASO Automotive Use Case and the distribution of cryptographic material across the different trust domains.

The main components involved in the TO-BE scenario are:

- Vehicle ECUs and Zonal Controllers: unchanged internal vehicle components responsible for vehicle functions and in-vehicle communications.
- In-Vehicle Network: existing vehicle communication infrastructure interconnecting ECUs, zonal controllers, and the OBU.
- On-Board Unit (OBU): vehicle gateway hosting the PiQASO SDK for external communications and integration with PQC-enabled services.
- PiQASO SDK: client-side integration layer connecting automotive and roadside applications with PQC-enabled cryptographic services and supporting trust infrastructures.
- PiQASO PQC-as-a-Service: external service platform providing PQC capabilities to consuming applications through the PiQASO integration framework.

- **FPGA-based Accelerator:** hardware component supporting accelerated execution of PQC operations such as ML-DSA and ML-KEM.
- **Roadside Unit (RSU):** infrastructure component enhanced with PiQASO components to support PQC-enabled V2I communications.
- **Backend Services:** external systems supporting traffic management, operational services, and future software update workflows.
- **Managed PQC-enabled vPKI:** trust infrastructure supporting certificate lifecycle management and PQC-enabled trust establishment.

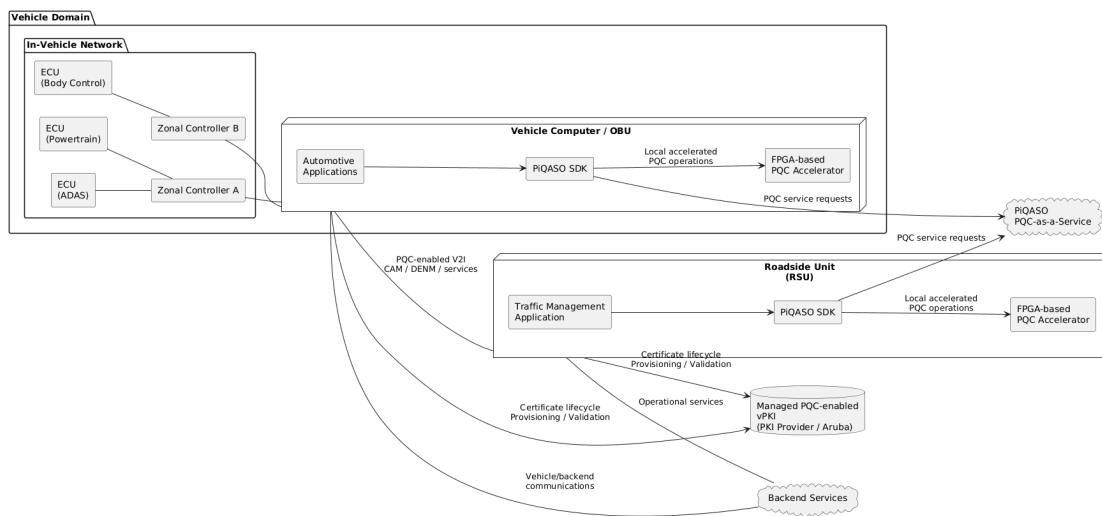


Figure 5.8: Automotive TO-BE System Architecture

Figure 5.8 illustrates the target architecture adopted in the PiQASO Automotive Use Case. Compared to the AS-IS scenario, the architecture introduces a dedicated integration layer based on the PiQASO SDK, local FPGA-based acceleration resources on both the OBU and RSU platforms, and a managed PQC-enabled vPKI supporting certificate lifecycle management and trust services. The architecture preserves the existing vehicle communication model while enabling the adoption of quantum-resistant cryptographic mechanisms at the communication boundary between vehicles, roadside infrastructure, backend services, and trust infrastructures.

This approach allows automotive applications to benefit from PQC-enabled services without requiring direct interaction with the underlying cryptographic implementations. At the same time, the use of managed trust services and hardware acceleration supports the operational, interoperability, and performance requirements of future V2X ecosystems. The following section describes the communication workflow implemented within the TO-BE architecture and the interaction between the different components during certificate provisioning, authentication, secure key establishment, and V2X message dissemination.

5.5.2 “To-Be” Reference Scenario #1: Quantum-Secure Emergency Vehicle Priority Management

The primary TO-BE reference scenario considered within the PiQASO Automotive Use Case addresses the secure interaction between an emergency vehicle and roadside infrastructure to enable traffic-light prioritisation through quantum-resistant V2I communications. The scenario extends the classical V2X

security architecture by introducing post-quantum cryptographic mechanisms for certificate provisioning, authentication, secure key establishment, and trusted message dissemination.

The proposed architecture combines the PiQASO SDK, managed PQC-enabled vPKI services, and optional hardware acceleration resources to support the complete lifecycle of secure communications between vehicles and roadside infrastructure. Particular attention is devoted to maintaining compatibility with existing V2X communication models while evaluating the operational impact introduced by PQC mechanisms.

The reference scenario is organised into two complementary phases. The first phase establishes the trust foundation through certificate provisioning and validation services provided by the managed PQC-enabled vPKI. The second phase enables trusted V2I communications through PQC-based authentication, quantum-resistant key establishment, and authenticated dissemination of safety-critical V2X messages.

Phase 1: PQC-Enabled Certificate Provisioning and Validation. Before participating in V2X communications, both vehicles and roadside infrastructure must obtain valid certificates issued by the managed PQC-enabled vPKI. In the proposed architecture, certificate requests and validation procedures are protected using post-quantum digital signatures generated through the PiQASO SDK; hardware acceleration may be used at implementation level as described in the target architecture.

The OBU interacts directly with the managed vPKI to authenticate itself, validate trust relationships, and periodically obtain pools of short-term pseudonym certificates together with updated revocation information. Local hardware acceleration resources may support the generation and verification of PQC signatures, reducing the computational impact associated with post-quantum algorithms.

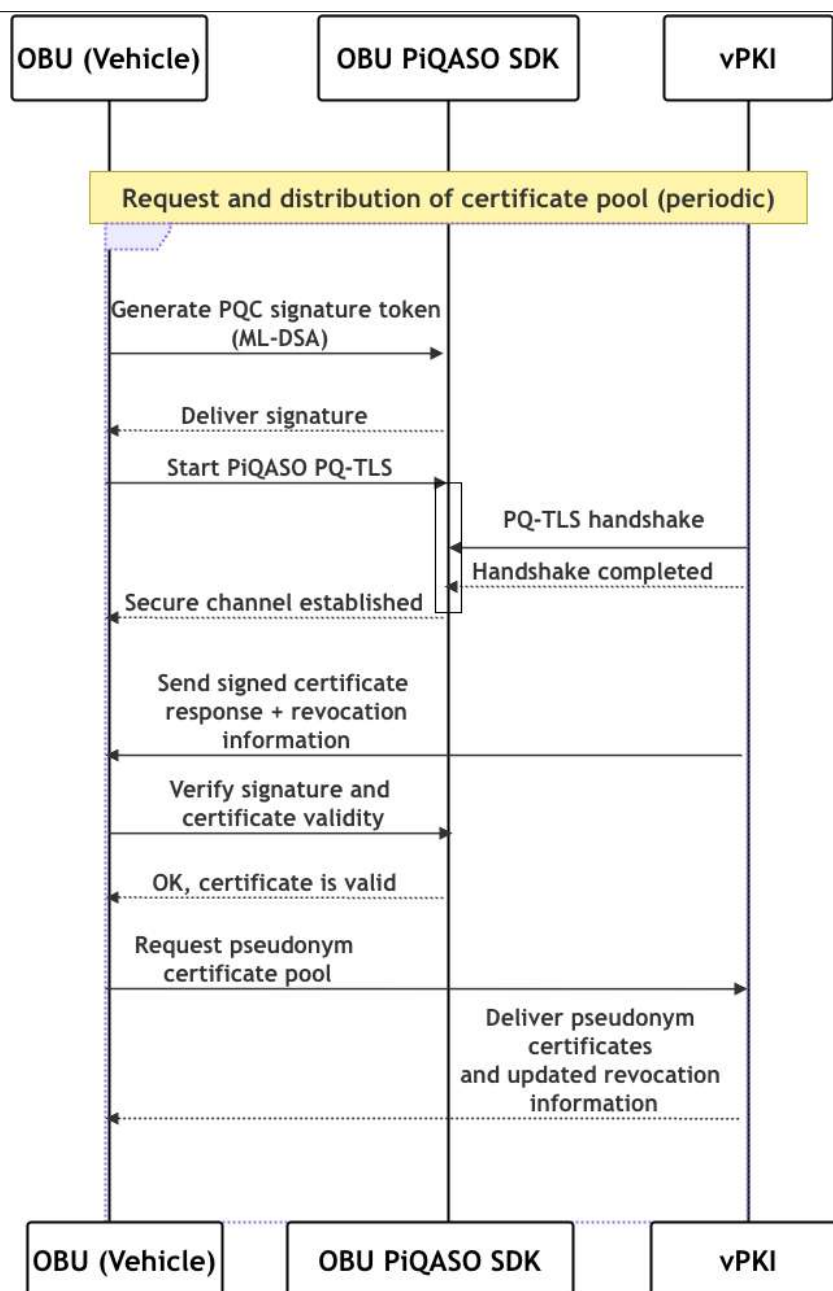


Figure 5.9: Automotive TO-BE Sequence Diagram - Certificate Provisioning and Validation

The certificate provisioning process establishes the trust foundation required for all subsequent communication activities. Figure 5.9 illustrates the certificate provisioning and validation workflow adopted in the TO-BE architecture.

The following steps describe the periodic request and distribution of the certificate pool between the On-Board Unit (OBU), the PiQASO SDK, and the vehicular Public Key Infrastructure (vPKI), as illustrated in the Figure.

1. **Signature token generation.** The OBU within the vehicle requests the PiQASO SDK to generate a post-quantum cryptographic (PQC) signature token using the ML-DSA algorithm.
2. **Signature delivery.** The PiQASO SDK returns the generated ML-DSA signature to the OBU.
3. **PQ-TLS session initiation.** The OBU instructs the PiQASO SDK to start a PiQASO PQ-TLS session with the vPKI.

4. **PQ-TLS handshake.** The PiQASO SDK performs the PQ-TLS handshake with the vPKI, establishing a quantum-resistant secure channel.
5. **Handshake completion.** The vPKI confirms completion of the handshake to the PiQASO SDK.
6. **Secure channel confirmation.** The PiQASO SDK notifies the OBU that the secure channel has been successfully established.
7. **Certificate response delivery.** The vPKI sends the signed certificate response together with the associated revocation information directly to the OBU over the established secure channel.
8. **Signature and certificate verification.** The OBU requests the PiQASO SDK to verify the signature and validate the received certificate.
9. **Validation result.** The PiQASO SDK confirms to the OBU that the certificate is valid.
10. **Pseudonym certificate pool request.** The OBU requests a pool of pseudonym certificates from the vPKI.
11. **Pseudonym certificate pool delivery.** The vPKI delivers the pseudonym certificates along with updated revocation information to the OBU, completing the periodic certificate provisioning cycle.

Phase 2: PQC Authentication, Secure Key Establishment and V2X Message Communication. Once valid certificates have been provisioned, OBUs and RSUs can establish trusted communication relationships using quantum-resistant authentication mechanisms. In the proposed architecture, both entities rely on the PiQASO SDK to perform post-quantum cryptographic operations while continuing to interact directly with the managed PQC-enabled vPKI for certificate status verification and trust validation.

After certificate provisioning, the communicating entities establish a PQC-protected communication channel through PiQASO PQ-TLS. The details of the PQ-TLS handshake are abstracted from this use case and described in a dedicated workflow common to all PiQASO use cases.

Following authentication, the communicating entities establish a shared secret using the ML-KEM key encapsulation mechanism. The OBU generates an ML-KEM public key and shares it with the RSU, which performs the encapsulation procedure and returns the resulting ciphertext. The OBU subsequently performs decapsulation and derives the same shared secret. This process enables the establishment of a quantum-resistant foundation for confidential communications without exposing the secret material over the communication channel.

The shared secret is then used to derive symmetric session keys that protect confidential unicast communications between vehicles and roadside infrastructure. This approach combines the efficiency of symmetric cryptography with the quantum-resistant properties of ML-KEM, ensuring that confidentiality can be preserved even in the presence of future quantum-capable adversaries.

For broadcast V2X communications, the architecture follows the same operational principles adopted in current C-ITS deployments. Cooperative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs) continue to be transmitted in plaintext to allow reception by all nearby participants. However, message authenticity and integrity are now protected through ML-DSA signatures generated via the PiQASO SDK. Receiving entities verify the signatures and certificate status before accepting the messages as valid and trustworthy.

5.5.3 "To-Be" Reference Scenario #2: Quantum-Secure OTA Software Update

Beyond V2I communications, modern connected vehicles increasingly rely on Over-The-Air (OTA) software update mechanisms to deploy firmware upgrades, security patches, and functional enhancements throughout the vehicle lifecycle. As vehicles remain operational for many years, the authenticity and integrity of software updates become critical security requirements. Compromised update mechanisms could allow attackers to distribute malicious software, alter vehicle functionality, or disrupt safety-critical services.

The second TO-BE reference scenario introduces a quantum-secure OTA software update workflow based on PQC-enabled authentication and post-quantum digital signature verification. The scenario demonstrates how the PiQASO architecture can be extended beyond V2I communications to support trusted software lifecycle management, while reusing the same managed PQC-enabled trust infrastructure and PiQASO SDK integration model adopted for vehicle-to-infrastructure interactions.

In the proposed workflow, the OBU authenticates towards the OTA Update Server using PQC-enabled credentials and an ML-DSA signature generated through the PiQASO SDK. The OTA Update Server validates the certificate and trust status through the managed PQC-enabled vPKI before accepting the update request. Once authentication has been completed, firmware metadata and firmware packages are exchanged over a protected PQ-TLS or hybrid TLS channel.

The OTA scenario also includes a quantum-resistant key-establishment phase based on ML-KEM. Following successful authentication, the OBU generates an ML-KEM key pair and provides the public key to the OTA infrastructure. The OTA server performs the encapsulation procedure and returns the resulting ciphertext, which is subsequently decapsulated by the OBU to derive a shared secret. This process establishes a PQC-protected communication context for update delivery and enables the validation of ML-KEM integration within the PiQASO OTA architecture.

Firmware authenticity and integrity are ensured through post-quantum digital signatures. The OTA infrastructure provides signed firmware metadata and signed firmware packages, which are verified by the OBU through the PiQASO SDK before installation is authorised. Only firmware packages successfully validated against the established trust chain and associated signature verification procedures are accepted and applied to the vehicle platform. Packages presenting invalid signatures, failed integrity checks, or untrusted provenance are rejected.

5.6 Reference Operational Scenarios

The validation activities of the PiQASO Automotive Use Case are based on a set of representative operational scenarios designed to assess the applicability of post-quantum cryptographic mechanisms within realistic automotive environments. These scenarios demonstrate how the architecture described in Section 5.5 can support secure V2I communications, trusted software lifecycle management, and cooperative mobility services while preserving the operational requirements of modern C-ITS ecosystems.

The primary validation scenarios focus on emergency vehicle priority management and quantum-secure OTA software updates, while an additional complementary scenario is considered to demonstrate the applicability of the proposed architecture to broader traffic safety services.

5.6.1 User Story #1 (Reference Scenario #1): Emergency Vehicle Priority Management

UC1.US1: Secure Emergency Vehicle Priority Management

As a Roadside Infrastructure Operator, I want to securely authenticate emergency vehicles requesting traffic-light prioritisation before granting priority access so that emergency vehicles can be reliably prioritised at traffic lights, while unauthorised parties are prevented from accessing priority management functions.

The primary validation scenario for the PiQASO Automotive Use Case is Emergency Vehicle Priority Management. The scenario addresses the secure interaction between an emergency vehicle and roadside infrastructure to enable traffic-light prioritisation while preventing unauthorised access to priority management functions.

The scenario involves the following actors:

- Emergency Vehicle equipped with an OBU, PiQASO SDK, and credentials issued by the managed PQC-enabled vPKI.
- Roadside Unit (RSU) responsible for receiving, validating, and processing priority requests from authorised emergency vehicles.
- Managed PQC-enabled vPKI providing certificate lifecycle management and trust services.

As the emergency vehicle approaches a signalised intersection, the OBU establishes a trusted communication relationship with the RSU using the PQC-enabled workflow described in Section 5.5. The vehicle authenticates itself through certificates issued by the managed vPKI and proves possession of valid credentials through ML-DSA-based authentication procedures.

Following successful authentication, the vehicle periodically transmits signed Cooperative Awareness Messages (CAMs) containing position, speed, direction of travel, and emergency-priority information. The RSU verifies the received signatures, validates the certificate status, and confirms the legitimacy of the sender before processing the priority request.

Once the request has been successfully validated, the RSU triggers the traffic-light prioritisation procedure according to the configured operational policies. This may include granting priority passage to the approaching emergency vehicle, extending green-light phases, or coordinating intersection behaviour to reduce delays and improve response times.

The use of PQC-enabled authentication mechanisms ensures that only authorised emergency vehicles can request traffic priority. Attempts to impersonate emergency services, inject fraudulent priority requests, or replay previously captured messages are detected and rejected through certificate validation and signature verification procedures.

The scenario is specifically designed to validate the following capabilities of the PiQASO architecture:

- PQC-enabled certificate provisioning and validation through the managed vPKI.
- ML-DSA-based authentication of vehicles and infrastructure.
- ML-KEM-based secure key establishment for protected communications.
- Integrity and authenticity protection of CAM messages.
- Resistance against unauthorised priority requests and vehicle impersonation attempts.

- Compliance with the latency requirements imposed by real-time V2I applications.

This scenario represents the primary reference environment for evaluating the effectiveness of quantum-resistant security mechanisms within operational V2I ecosystems.

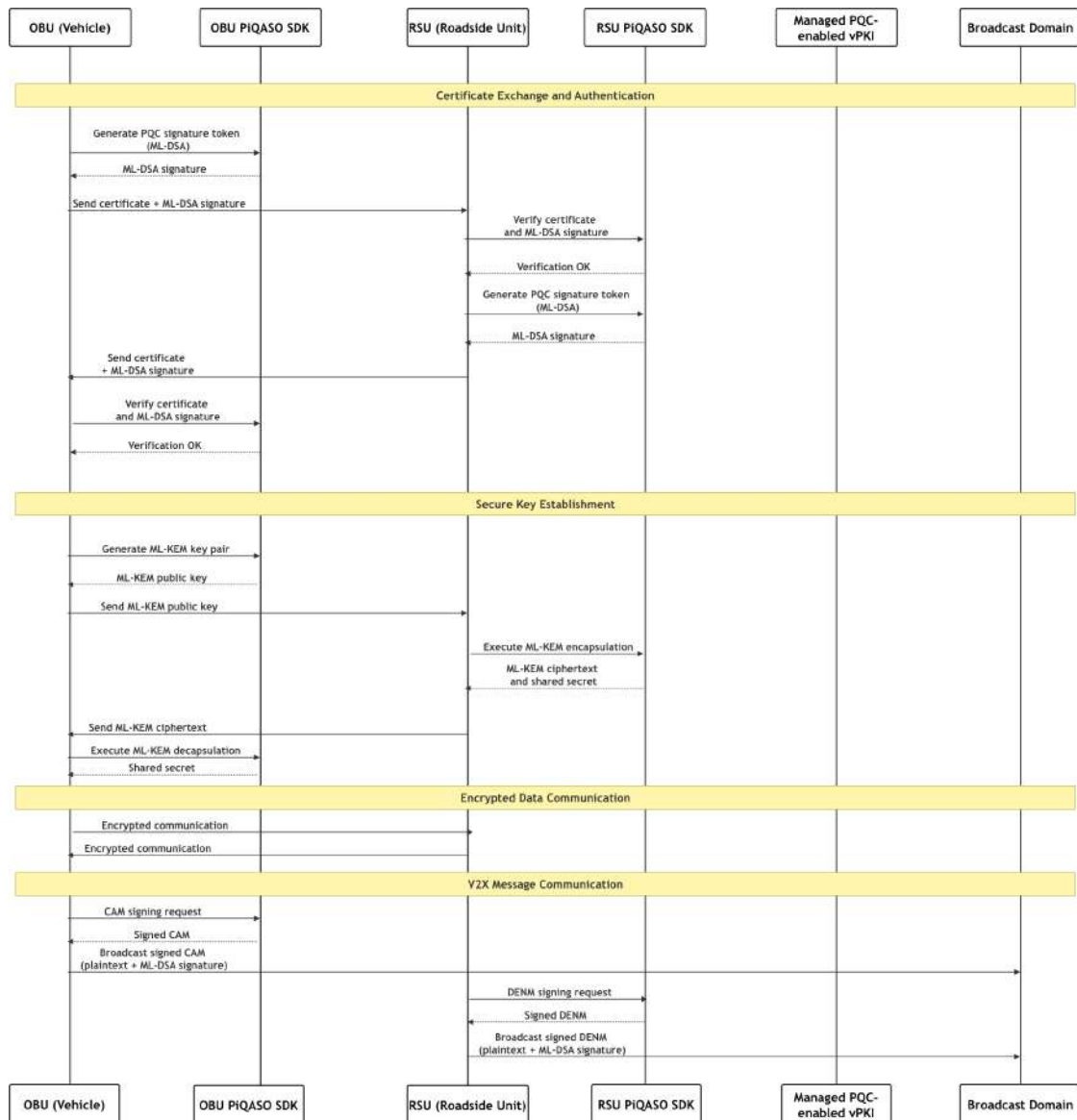


Figure 5.10: Automotive TO-BE Sequence Diagram - Auth, Secure Key and V2X Message Communication

Figure 5.10 illustrates the complete TO-BE communication workflow, covering mutual authentication, quantum-resistant key establishment, protected unicast communications, and authenticated V2X message dissemination. Together, the two phases demonstrate how post-quantum cryptographic mechanisms can be integrated into emergency vehicle priority management services while preserving the operational principles of existing C-ITS deployments.

The following steps describe the mutual certificate-based authentication between an OBU and an RSU, the subsequent establishment of a shared secret via ML-KEM, and the signing and broadcasting of V2X messages, as illustrated in the Figure.

Phase 1: Certificate exchange and authentication.

1. **OBU signature token generation.** The OBU requests the OBU PiQASO SDK to generate a PQC signature token using ML-DSA.
2. **OBU signature delivery.** The OBU PiQASO SDK returns the ML-DSA signature to the OBU.
3. **OBU certificate transmission.** The OBU sends its certificate together with the ML-DSA signature to the RSU.
4. **OBU certificate verification.** The RSU requests the RSU PiQASO SDK to verify the received certificate and ML-DSA signature.
5. **OBU verification confirmation.** The RSU PiQASO SDK confirms to the RSU that verification succeeded.
6. **RSU signature token generation.** The RSU requests the RSU PiQASO SDK to generate its own PQC signature token using ML-DSA, in order to authenticate itself back to the OBU.
7. **RSU signature delivery.** The RSU PiQASO SDK returns the ML-DSA signature to the RSU.
8. **RSU certificate transmission.** The RSU sends its certificate together with the ML-DSA signature to the OBU, completing the mutual exchange.
9. **RSU certificate verification.** The OBU requests the OBU PiQASO SDK to verify the RSU's certificate and ML-DSA signature.
10. **RSU verification confirmation.** The OBU PiQASO SDK confirms to the OBU that verification succeeded, completing mutual authentication between the two parties.

Phase 2: Secure key establishment.

11. **ML-KEM key pair generation.** The OBU requests the OBU PiQASO SDK to generate an ML-KEM key pair.
12. **Public key delivery.** The OBU PiQASO SDK returns the ML-KEM public key to the OBU.
13. **Public key transmission.** The OBU sends the ML-KEM public key to the RSU.
14. **Encapsulation execution.** The RSU requests the RSU PiQASO SDK to execute ML-KEM encapsulation against the received public key.
15. **Ciphertext and secret delivery.** The RSU PiQASO SDK returns the resulting ML-KEM ciphertext and the derived shared secret to the RSU.
16. **Ciphertext transmission.** The RSU sends the ML-KEM ciphertext to the OBU.
17. **Decapsulation execution.** The OBU requests the OBU PiQASO SDK to execute ML-KEM decapsulation on the received ciphertext.
18. **Shared secret derivation.** The OBU PiQASO SDK returns the derived shared secret to the OBU, completing key establishment: both the OBU and the RSU now hold a matching shared secret without it having been transmitted directly.

Phase 3: Encrypted data communication.

19. **Uplink encrypted exchange.** The OBU sends encrypted communication, protected under the shared secret established via ML-KEM, to the RSU.
20. **Downlink encrypted exchange.** The RSU responds with encrypted communication to the OBU, confirming that the bidirectional secure channel is operational.

Phase 4: V2X message communication.

21. **CAM signing request.** The OBU requests the OBU PiQASO SDK to sign a Cooperative Awareness Message (CAM).
22. **CAM signature delivery.** The OBU PiQASO SDK returns the signed CAM to the OBU.
23. **CAM broadcast.** The OBU broadcasts the signed CAM, sent as plaintext accompanied by the ML-DSA signature, to the Broadcast Domain.
24. **DENM signing request.** The RSU requests the RSU PiQASO SDK to sign a Decentralized Environmental Notification Message (DENM).
25. **DENM signature delivery.** The RSU PiQASO SDK returns the signed DENM to the RSU.
26. **DENM broadcast.** The RSU broadcasts the signed DENM, sent as plaintext accompanied by the ML-DSA signature, to the Broadcast Domain, completing the V2X communication cycle.

5.6.2 User Story #2 (Reference Scenario #2): Quantum-Secure OTA Software Update

UC1.US2: Secure Distribution of Software and Firmware Updates to Connected Vehicles

As a Roadside Infrastructure Operator, I want to securely distribute software and firmware updates to connected vehicles while preserving software authenticity, integrity, and provenance throughout the update lifecycle so that vehicles are protected against the installation of unauthorised or malicious software.

The second operational scenario considered within the PiQASO Automotive Use Case addresses the secure distribution of software and firmware updates to connected vehicles. The scenario focuses on preserving software authenticity, integrity, and provenance throughout the update lifecycle, thereby protecting vehicles against the installation of unauthorised or malicious software.

The scenario involves the following actors:

- Vehicle equipped with an OBU, PiQASO SDK, and credentials issued by the managed PQC-enabled vPKI.
- OTA Update Server responsible for distributing firmware packages and software updates.
- Managed PQC-enabled vPKI providing certificate lifecycle management and trust services.

During normal operation, the vehicle periodically checks for available software updates. Following successful authentication, the vehicle and the OTA infrastructure establish a protected communication context through ML-KEM-based key establishment and PQ-TLS. Firmware metadata and software packages

are authenticated through PQC digital signatures before installation is authorised. Before accepting update information or downloading software packages, the vehicle establishes a trusted relationship with the OTA infrastructure and verifies the authenticity of the update source.

When a new software version becomes available, the OTA infrastructure provides update metadata describing the software package, version information, and associated validation data. The vehicle validates the received information before initiating the download process. Once the firmware package has been received, integrity and authenticity checks are performed prior to installation.

Only software packages successfully validated against the established trust chain are accepted and applied. Any software package presenting invalid signatures, failed integrity checks, or untrusted provenance is rejected.

This scenario validates the capability of the PiQASO architecture to support trusted software lifecycle management processes while extending the same PQC-enabled trust infrastructure already used for V2I communications. In particular, it demonstrates how quantum-resistant authentication and trust services can contribute to the long-term security of software-defined vehicles.

The validation activities will primarily rely on ML-DSA as the baseline signature mechanism for firmware metadata and firmware package authentication. In addition, the OTA scenario will be used to evaluate alternative PQC signature primitives, including FALCON and other candidate algorithms offering reduced signature-verification latency. The objective is to compare their impact on signature size, verification performance, and OTA update validation time while maintaining the same overall trust model.

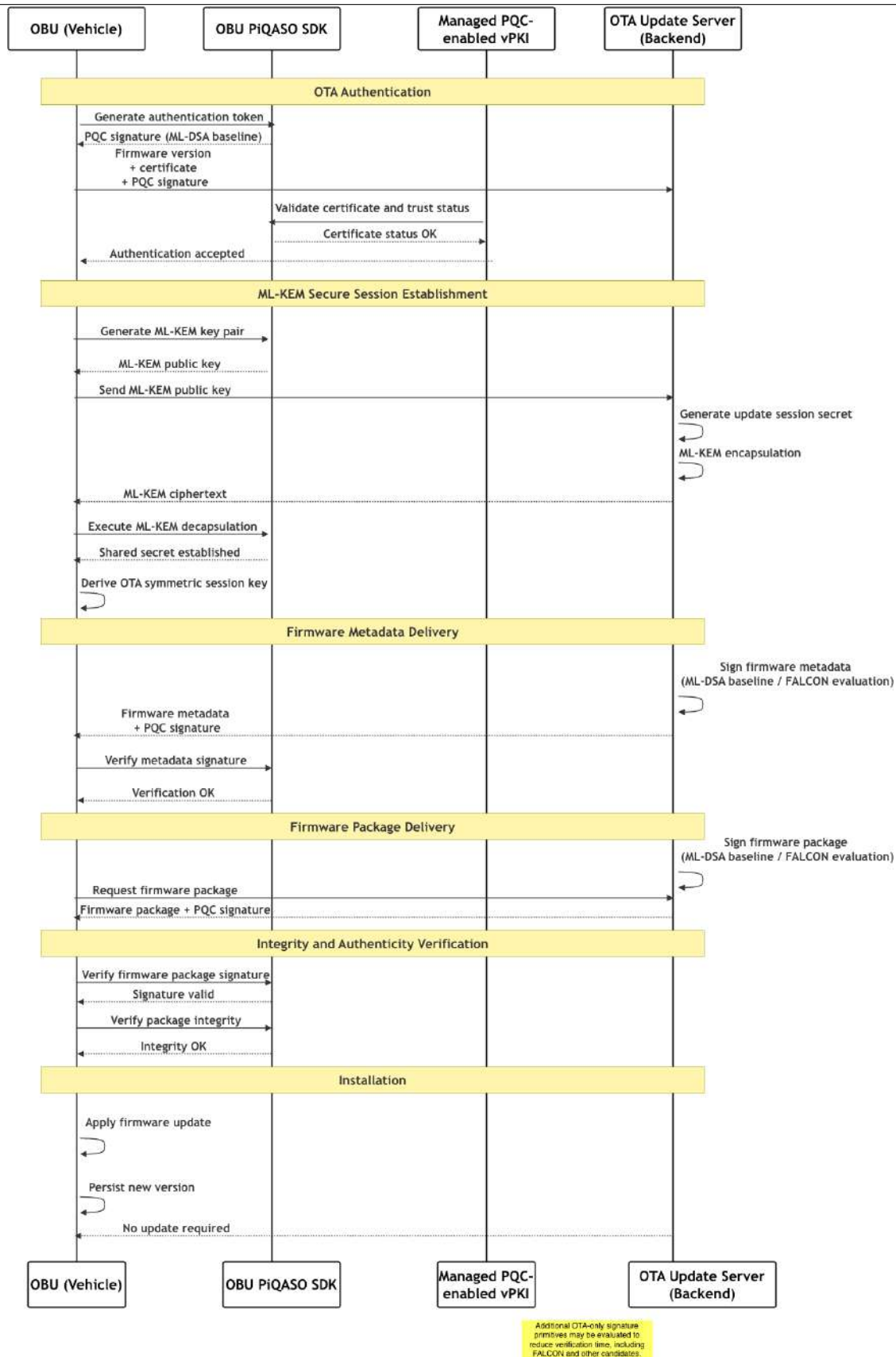


Figure 5.11: Automotive TO-BE Sequence Diagram - Quantum-Secure OTA Update

Figure 5.11 illustrates the quantum-secure OTA update workflow implemented within the PiQASO Automotive Use Case. The workflow covers firmware update checking, OBU authentication, certificate validation through the managed PQC-enabled vPKI, firmware metadata verification, firmware package

download, signature and integrity validation, and final installation of the authenticated update.

The following steps describe the end-to-end quantum-resistant OTA firmware update procedure between the OBU, the OBU PiQASO SDK, the Managed PQC-enabled vPKI, and the OTA Update Server, as illustrated in the Figure.

Phase 1: OTA authentication.

1. **Authentication token generation.** The OBU requests the OBU PiQASO SDK to generate an authentication token.
2. **PQC signature delivery.** The OBU PiQASO SDK returns a PQC signature, using the ML-DSA baseline, to the OBU.
3. **Authentication request transmission.** The OBU sends its firmware version, certificate, and PQC signature to the OTA Update Server.
4. **Certificate validation.** The OTA Update Server requests the Managed PQC-enabled vPKI to validate the certificate and its trust status.
5. **Certificate status confirmation.** The Managed PQC-enabled vPKI confirms to the OTA Update Server that the certificate status is valid.
6. **Authentication acceptance.** The OTA Update Server notifies the OBU that authentication has been accepted, completing the initial trust establishment.

Phase 2: ML-KEM secure session establishment.

7. **ML-KEM key pair generation.** The OBU requests the OBU PiQASO SDK to generate an ML-KEM key pair.
8. **Public key delivery.** The OBU PiQASO SDK returns the ML-KEM public key to the OBU.
9. **Public key transmission.** The OBU sends the ML-KEM public key to the OTA Update Server.
10. **Session secret generation.** The OTA Update Server internally generates the update session secret.
11. **Encapsulation.** The OTA Update Server internally performs ML-KEM encapsulation against the received public key.
12. **Ciphertext delivery.** The OTA Update Server returns the resulting ML-KEM ciphertext to the OBU.
13. **Decapsulation execution.** The OBU requests the OBU PiQASO SDK to execute ML-KEM decapsulation on the received ciphertext.
14. **Shared secret confirmation.** The OBU PiQASO SDK confirms to the OBU that the shared secret has been established.
15. **Session key derivation.** The OBU internally derives the OTA symmetric session key from the shared secret, completing secure session establishment.

Phase 3: Firmware metadata delivery.

16. **Metadata signing.** The OTA Update Server internally signs the firmware metadata, using the ML-DSA baseline with FALCON under evaluation as an alternative.

17. **Metadata transmission.** The OTA Update Server sends the firmware metadata together with the PQC signature to the OBU.
18. **Metadata signature verification.** The OBU requests the OBU PiQASO SDK to verify the metadata signature.
19. **Verification confirmation.** The OBU PiQASO SDK confirms to the OBU that verification succeeded.

Phase 4: Firmware package delivery.

20. **Package signing.** The OTA Update Server internally signs the firmware package, using the ML-DSA baseline with FALCON under evaluation as an alternative.
21. **Package request.** The OBU requests the firmware package from the OTA Update Server.
22. **Package delivery.** The OTA Update Server returns the firmware package together with the PQC signature to the OBU.

Phase 5: Integrity and authenticity verification.

23. **Package signature verification.** The OBU requests the OBU PiQASO SDK to verify the firmware package signature.
24. **Signature validity confirmation.** The OBU PiQASO SDK confirms to the OBU that the signature is valid.
25. **Package integrity verification.** The OBU requests the OBU PiQASO SDK to verify the integrity of the firmware package.
26. **Integrity confirmation.** The OBU PiQASO SDK confirms to the OBU that integrity verification succeeded, clearing the package for installation.

Phase 6: Installation.

27. **Firmware update application.** The OBU internally applies the verified firmware update.
28. **Version persistence.** The OBU internally persists the new firmware version.
29. **Update status confirmation.** The OTA Update Server confirms to the OBU that no further update is required, concluding the OTA update cycle.

Note: Additional OTA-specific signature primitives may be evaluated to reduce verification time, including FN-DSA and other post-quantum signature candidates.

5.6.3 User Story #3 (Reference Scenario #3): Complementary Scenario: Intersection Safety Coordination

UC1.US3: Authenticated V2I Communication for Intersection Safety Coordination

As a Roadside Infrastructure Operator, I want to provide an intersection safety coordination service based on authenticated V2I (Vehicle-to-Infrastructure) communications with vehicles so that the proposed security architecture is validated across a broader set of cooperative mobility services, beyond the primary emergency vehicle priority management scenario.

As a complementary validation scenario, the PiQASO Automotive Use Case considers an intersection safety coordination service based on authenticated V2I communications between vehicles and roadside infrastructure. While the emergency vehicle priority management scenario remains the primary validation target, this additional scenario demonstrates the applicability of the proposed architecture to a broader set of cooperative mobility services.

In this scenario, vehicles approaching a signalised intersection periodically transmit signed Cooperative Awareness Messages (CAMs) containing position, speed, heading, and other situational information. The RSU collects and validates the received messages using the managed PQC-enabled trust infrastructure and evaluates the traffic situation within its coverage area.

Based on the received information, the RSU may generate and disseminate Decentralized Environmental Notification Messages (DENMs) to inform approaching vehicles about potentially hazardous conditions, intersection occupancy, traffic conflicts, or recommended driving actions. The authenticity and integrity of these notifications are protected through PQC-enabled digital signatures, ensuring that only trusted infrastructure components can generate safety-related information.

The scenario validates the capability of the proposed architecture to support secure bidirectional V2I communications, trusted message dissemination, and quantum-resistant authentication mechanisms within safety-critical traffic environments. In particular, it demonstrates how the same certificate lifecycle management, authentication procedures, and cryptographic services adopted for emergency vehicle prioritisation can be reused for cooperative traffic safety applications without requiring architectural modifications.

5.7 Validation Strategy, Evaluation Metrics and Acceptance Criteria

The validation activities of the PiQASO Automotive Use Case are designed to assess both the technical feasibility of integrating Post-Quantum Cryptography within automotive communication ecosystems and the operational impact of the proposed architecture on representative automotive services. Particular attention is devoted to evaluating quantum-resistant V2I communications, trusted software lifecycle management, and the supporting trust infrastructure required to secure future software-defined vehicles. Validation activities will be conducted using the reference architecture described in Section 5.5, involving OBU and RSU platforms equipped with the PiQASO SDK, a managed PQC-enabled vPKI, and the supporting communication infrastructure required to execute the operational scenarios described in Section 5.6. The validation campaign will cover certificate provisioning, mutual authentication, secure key establishment, authenticated V2X message dissemination, and quantum-secure OTA software update procedures.

The validation strategy combines functional verification, interoperability testing, and performance evaluation. Functional tests will verify the correct execution of certificate lifecycle operations, authentication

procedures, secure communication establishment, message signing and verification, and OTA update validation workflows. Interoperability tests will assess the interaction between vehicles, roadside infrastructure, PiQASO components, and the managed vPKI. Performance tests will evaluate the impact of PQC mechanisms on communication latency, system responsiveness, and operational scalability.

Table 5.2: Performance and Security KPIs for the Automotive Use Case

KPI	Description	Acceptance Criteria
CAM/DENM Processing Budget Compliance	Ability of the system to process authenticated CAM and DENM messages while respecting ETSI timing constraints for V2X communications.	Per-message OBU $\leftrightarrow$ RSU processing cycle $\leq$ 250 ms.
Emergency Vehicle Priority Request Handling	Ability of the RSU to receive, authenticate, and process emergency vehicle priority requests without impacting operational traffic management functions.	Priority requests successfully processed within the CAM/DENM processing budget.
RSU Concurrent Vehicle Capacity	Number of vehicles simultaneously supported by the RSU while maintaining compliant operation.	$\geq$ 50 vehicles simultaneously served at a 10 Hz message rate.
Pseudonym Rotation Transparency	Ability to rotate pseudonym certificates without disrupting V2X communications.	No CAM/DENM message loss during certificate rotation; first post-rotation message accepted without re-authentication procedures.
OBU Service Availability After Startup	Time required for a vehicle to become operational and transmit authenticated V2X messages after startup.	First authenticated CAM transmitted within 50 ms from service startup.
Secure Communication Establishment	Ability to successfully establish authenticated and protected communications between OBU and RSU using PQC mechanisms.	100% successful completion during validation activities.
OTA Update Authenticity Verification	Ability to verify the authenticity and integrity of OTA software updates before installation.	100% acceptance of valid firmware packages and rejection of tampered firmware packages.
OTA Update Service Continuity	Ability to complete the OTA workflow while preserving trust validation procedures and software integrity checks.	Successful end-to-end OTA update execution during validation activities.

Table 5.2 summarises the Key Performance Indicators (KPIs) and acceptance criteria defined for the Automotive Use Case.

In addition to the use-case-driven KPIs listed above, a set of technical evaluation metrics will be collected during validation activities. These include ML-DSA signature generation and verification latency, ML-KEM key establishment latency, certificate provisioning latency, communication overhead introduced by PQC artefacts, and the performance benefits achieved through hardware acceleration. These measurements will provide quantitative evidence supporting the assessment of the proposed architecture and will contribute to the definition of migration strategies for future quantum-secure automotive ecosystems.

The successful completion of the validation activities and the achievement of the defined acceptance criteria will demonstrate the feasibility of integrating PQC technologies into automotive V2X environments while preserving the operational requirements of real-world intelligent transportation systems.

Chapter 6

PiQASO UC2 – Sector: Automation

6.1 Motivation and Challenges in the Automation Sector

6.1.1 Current Landscape: Autonomous Mobile Robots in Industrial Deployments

The global intralogistics sector—encompassing autonomous material handling within warehouses, production facilities, and distribution centers—is experiencing rapid adoption of **Autonomous Mobile Robots (AMRs)**. This trend is particularly evident across the *e-commerce, logistics, automotive, healthcare, and electronics sectors, where warehouse automation, line feeding, and collaborative material handling have emerged as primary focus areas*. Consequently, major global operators have moved beyond pilot programs to integrate hundreds of thousands of units across their logistics networks, fundamentally transforming fulfilment speed and supply chain resilience [9, 68]. Industry platforms such as **TIAGo OMNI Base** (PAL Robotics), **MiR Fleet** (Mobile Industrial Robots), and **Clearpath Jackal** represent the current state-of-the-art in real-time autonomous navigation and fleet coordination.

The operational backbone of these deployments relies on centralized orchestration frameworks (e.g., **Open-RMF**) that facilitate high-frequency data exchange via **ROS 2** middleware. While ROS 2 was designed to overcome the inherent security lacunae of its predecessor by leveraging the **Data Distribution Service (DDS)** standard, a significant “security-usability gap” remains. Current production environments predominantly lean on legacy network-layer defences—such as VPN overlays (WireGuard) and firewalled perimeters—under the assumption of a “trusted intranet.”

However, empirical evidence suggests that this perimeter-based approach is becoming increasingly inadequate. Research by **Alias Robotics** has demonstrated that the DDS middleware layer remains a primary attack vector; vulnerabilities in widely used implementations (e.g., eProsima Fast DDS, RTI Connext) can lead to Denial of Service (DoS) or Remote Code Execution (RCE), potentially paralyzing entire fleets [5]. Despite the existence of **SROS2** and the **DDS-Security** specifications (providing authentication, access control, and encryption), these features are frequently bypassed due to their configuration complexity and perceived performance overhead.

The cybersecurity posture of existing AMR deployments is further complicated by critical flaws in the underlying robot controllers. CISA (Cybersecurity and Infrastructure Security Agency) has documented severe vulnerabilities (e.g., **ICSA-18-191-01**) affecting major vendors like Universal Robots, where missing authentication for critical functions (Ports 30001-30003) allows unauthenticated URScript execution with root privileges [20]. Similarly, ICS-ALERT-20-217-01 highlights that “robot motion servers” in multiple OEM platforms are susceptible to RCE, enabling attackers to exfiltrate data or disrupt physical movements.

Consequently, many industrial AMRs remain “secure at the edge but vulnerable at the core,” leaving

critical ROS 2 topics exposed to unauthorized eavesdropping, spoofing, and replay attacks within the internal network. Simultaneously, the robotics community is beginning to address the long-term threat of quantum computing, recognizing that cryptographic agility—the ability to transition from classical to post-quantum algorithms—will become an operational necessity to ensure the integrity of robotic fleets throughout their multi-year lifecycle.

6.1.2 The Convergence of Cybersecurity and Physical Safety

In the automation sector, the primary challenge lies in the deep convergence between **Cybersecurity** and **Physical Safety**. Unlike traditional IT systems, where a security breach typically results in data exfiltration or financial loss, a compromise in a deployed robotic fleet has immediate, measurable, and potentially lethal physical consequences. Since AMRs operate in “shared spaces” alongside human workers and stationary infrastructure, the boundary between digital integrity and operational safety is virtually non-existent.

- **Safety Incidents:** A malicious actor exploiting unencrypted or unauthenticated ROS 2 topics could inject falsified velocity commands, disable obstacle detection sensors, or corrupt environmental maps, causing collisions with personnel or equipment. *Documented robotic incidents have demonstrated that even minor control signal tampering can result in physical injury or fatalities in shared industrial workspaces [76].*
- **Operational Disruption:** Spoofed or replay-attacked mission commands could cause robots to abandon their assigned tasks, leading to production halts and significant operational losses. *Even a minor disruption in fleet coordination can lead to production halts in facilities relying on 24/7 autonomous operations.*
- **Data Exfiltration:** Unencrypted rosbags and facility maps transmitted over the network could expose sensitive layouts and proprietary operational logic to industrial competitors or nation-state adversaries.

Maintaining the integrity and authenticity of every control signal and navigation command is therefore not merely a matter of data protection, but a **fundamental safety requirement** that directly impacts the well-being of human workers and the operational continuity of critical industrial processes.

6.1.3 The Transition from Classical to Post-Quantum Cryptography in Robotics

The broader cybersecurity landscape is undergoing a paradigm shift toward post-quantum cryptography (PQC) readiness. *On August 13, 2024, NIST released the first three finalized Post-Quantum Cryptography standards [76]* to help critical infrastructures transition away from classical public-key algorithms that quantum computers could compromise. However, despite NIST’s PQC standardization, the robotics and autonomous systems community has had limited engagement, with most production deployments not yet implementing quantum-resistant cryptography. While some industry initiatives have begun exploring PQC integration in robotic systems, there remains a gap in practical guidance for deploying PQC in real-time robotic systems without sacrificing performance or safety guarantees.

The specific challenges for robotic systems are distinct from generic IT infrastructure:

- **Real-Time Constraints:** AMRs operate under strict latency budgets (tens of milliseconds for control loops). PQC algorithms introduce computational overhead that can affect real-time navigation

performance. Deploying PQC in high-frequency loops requires careful optimization and, in many cases, hardware acceleration [39].

- **Resource Constraints:** Fleet robots operate on embedded systems with limited CPU, memory, and battery capacity. PQC algorithms have larger key sizes and computational footprints than classical counterparts, raising practical questions about feasibility in resource-constrained environments [50].
- **Operational Agility:** Industrial operators must transition existing fleets to PQC without complete infrastructure overhauls. This requires **cryptographic agility**—the ability to support both classical and post-quantum algorithms simultaneously. Current robotics deployments lack such hybrid cryptography support [53].
- **Long-Term Data Protection:** The *"harvest now, decrypt later"* threat is particularly acute in robotics. Industrial facilities collect and retain operational logs, maps, and telemetry for months or years. Data intercepted today could be decrypted once quantum computers become available, exposing sensitive proprietary information [51].

Current industrial deployments lack any explicit support for post-quantum cryptography. Existing PKI infrastructures (based on RSA or ECDSA) remain the norm across deployed AMR fleets, and the transition pathway—both technically and organizationally—remains opaque to most operators.

6.1.4 Motivation for PiQASO's Role in the Automation Sector

The convergence of these challenges—cyber-physical safety criticality, real-time constraints, resource limitations, and the urgent need for PQC adoption—creates a compelling case for **designing and validating post-quantum secure fleet architectures in realistic operational environments**. The PiQASO project addresses this critical gap by:

- **PQC-Aware Fleet Protocols:** Demonstrating how to integrate post-quantum cryptographic functionalities—including PQ Signatures, PQ KEM, and PQ-TLS—into real-world robotic fleet communications, as detailed in Table 6.1. This bridges the gap between cryptographic security and real-time operational requirements.
- **Hardware Acceleration Solutions:** Utilizing **FPGA-based quantum-resistant trusted computing bases (QR-TCB)** to offload PQC operations, maintaining real-time performance guarantees while preserving cryptographic strength. This approach addresses the latency and power consumption challenges identified as key barriers to PQC adoption in embedded robotics.
- **Cryptographic Agility Frameworks:** Providing SDKs and services that enable seamless migration from classical to post-quantum algorithms, supporting mixed-mode deployments during transition periods. This reduces operational friction for industrial operators deploying PQC across legacy and new fleet infrastructure.
- **Technical Validation:** Evaluating PQC solutions on robotic platforms (TIAGo OMNI Base) in controlled laboratory settings at PAL Robotics, demonstrating practical feasibility and measurable performance characteristics under realistic operational scenarios. Laboratory validation of PQC integration in real robotic systems is essential to establish technical confidence in PQC adoption for robotics.

Through this use case, PiQASO contributes to accelerating post-quantum cryptography adoption in the automation and robotics sector, providing reference architectures, performance benchmarks, and lessons learned for the next generation of secure autonomous systems.

6.2 Use case's Description, Goals and Objectives

To address these domain-specific challenges, the second Use Case of PiQASO focuses on the practical deployment of Autonomous Mobile Robots (AMRs) in real-world intralogistics scenarios. In this setting, the focus is on a fleet of TIAGo OMNI Base robots that must navigate through dynamic warehouse floors while maintaining a high level of coordination.

In a typical intralogistics scenario, an "ego" AMR (TIAGo OMNI Base) navigating through a dynamic warehouse floor utilizes high-frequency ROS 2 messages to exchange navigation data—such as poses, velocity commands, and obstacle detections—with neighboring robots and the central infrastructure. This real-time information allows the AMR to predict the trajectories of other moving entities, identify potential collision zones in narrow aisles, and adjust its path dynamically to ensure safe and efficient movement. Simultaneously, the robot interacts with a central Fleet Manager (based on Open-RMF) to receive mission-critical instructions, such as task allocations (e.g., "deliver package to Station 5") and provides low-frequency status updates to ensure the entire fleet remains synchronized.

The arrival of quantum computers poses a severe risk to these fleets, since quantum adversaries could eventually compromise classical public-key algorithms, enabling *unauthorized robot control, disruption of fleet operations, or leakage of sensitive data such as maps and logs*. This would not only cause operational disruption but also safety incidents in shared spaces.

The goal of this Use Case is to **secure all AMR fleet communications using post-quantum cryptography (PQC)**, integrated via PiQASO's SDK and services. This is essential to evaluate and guarantee the trust and integrity of the data used in the robots' decision-making process before they are executed in the real-world environment.

To meet these challenges, the Automation use case explores reference scenarios where PiQASO's functionalities protect the fleet from quantum-capable adversaries. The primary objectives are:

- Ensure **end-to-end confidentiality, authenticity, and integrity** of fleet communications (navigation data and mission commands) against quantum-capable adversaries.
- Provide **PQC-secured identity and onboarding** for each robot before it joins the fleet.
- Introduce **fine-grained authorization** for fleet operations. , differentiating between basic telemetry access for operators and privileged access for critical commands (e.g., emergency stops or route overrides).
- Secure **data-at-rest** such as rosbags, maps, and logs with sealed PQC-protected storage.
- Benchmark PQC performance in robotic real-time loops, measuring latency, throughput, and resource overhead.

6.3 PiQASO Functionalities

In accordance with the project objectives, the Automation Use Case (PAL Robotics) will showcase the integration of specific PQC functionalities to secure fleet operations. The following table details these functionalities, their specific application within the AMR ecosystem, and the planned reporting period for their demonstration.

Table 6.1: PiQASO Functionalities to be evaluated for Use Case 2 (Automation)

PiQASO Functionality	Description and Purpose in the Use Case	RP1	RP2
PQ Signatures	Used by the Fleet Manager (FMA) to sign task assignments, ensuring mission commands are authentic and untamperable.	✓	
PQ-TLS (incorporating ML-KEM)	Establishes a secure communication channel between the FMA and the AMRs using quantum-resistant handshakes and session key encapsulation.	✓	
QR-TCB	Utilization of the Quantum-Resistant Trusted Computing Base to ensure a secure hardware-rooted execution environment for the fleet.	✓	✓
Hardware Acceleration	Offloading of PQC primitives to FPGA-based hardware to maintain real-time navigation performance under high-frequency loops.		✓
AKE	Establishment of ephemeral sessions through Authenticated Key Exchange for secure robot-to-robot interactions.		✓
UPKE	Direct encryption of operational data by the AMRs, allowing robots to protect telemetry before transmission or storage.		✓
ABE	Applied to operational logging to ensure that rosbags and maps are accessible only to users with specific roles (e.g., Admins).		✓

6.3.1 Rationale for the Phased Implementation

The deployment of these functionalities is structured to build a progressively more resilient security architecture. During the **first reporting period (RP1)**, the focus is on securing the core control flow and real-time command distribution using PQ-TLS (with ML-KEM) and Signatures, while establishing the foundation for hardware trust via the QR-TCB. The **second reporting period (RP2)** will focus on performance optimization and advanced security features, including the full deployment of FPGA-based hardware acceleration, the introduction of AKE for dynamic interactions, and the implementation of ABE and UPKE for data-at-rest protection and granular access control to logs and maps.

6.4 “As-is” Scenario

The current operational architecture for the AMR fleets of PAL Robotics is illustrated in Figure 6.1. It depicts a centralized coordination system where the security perimeter is enforced at the network level, but lacks application-specific and quantum-resistant protections.

Scenario Narrative

The “As-is” architecture is centered around a **Fleet Manager** based on the **Open-RMF** framework, which orchestrates tasks across a fleet of **TIAGO OMNI Base robots**. Each AMR runs the ROS2 Navigation2 stack within Docker containers to ensure the environment isolation and deployment consistency. However, as shown in the diagram, this setup presents several critical security gaps. The primary communication channel, based on ROS 2 middleware, currently operates in **plaintext** (no encryption/authentication). Although DDS security extensions exist and are theoretically supported by the middleware, they are not enabled in the current deployment. Consequently, all ROS 2 topics remain exposed, leaving high-frequency navigation data and mission-critical commands vulnerable to spoofing, replay attacks, and eavesdropping. An adversary capable of intercepting this traffic could inject malicious velocity commands or falsify obstacle detections, leading to potential safety incidents in shared workspaces.

To provide a security baseline, the system uses VPN overlays. Currently, **OpenVPN** with a **custom CA** (Certificate Authority) is deployed in production to isolate the fleet's network. To address the high-performance requirements of robotic systems, **WireGuard** is currently under evaluation as a more efficient alternative to traditional VPNs. Its implementation is being tested specifically for its superior throughput and lower latency, which are critical for maintaining real-time synchronization in high-frequency ROS 2 data loops. Additionally, WireGuard's **simplified keying mechanism** is being explored to streamline the deployment process and reduce the administrative overhead associated with managing a large-scale fleet of mobile robots. For maintenance and remote management, **SSH** is used to provide remote access, which is strictly layered on the VPN. Despite these protections, the current architecture faces significant security challenges. The VPN trust is **"flat"**, meaning that all peers on the network are trusted equally without application-layer separation. This allows a compromised peer to potentially "over-reach," accessing sensitive topics or services of the entire fleet.

Furthermore, the management of operational data remains a concern. Data collected for **Monitoring & Logs** via **rosbags**, **Grafana**, and **Prometheus** is stored unencrypted. This leaves sensitive industrial assets, such as high-resolution facility maps and operational logs, exposed to unauthorized access. Finally, the current cryptographic baseline relies exclusively on classical algorithms. There are currently no post-quantum authentication or robot attestation mechanisms in place, making the fleet vulnerable to future quantum adversaries who could compromise the long-term confidentiality of the collected data through "harvest now, decrypt later" strategies.

Summary of Key Vulnerabilities

- **Communication:** ROS 2 traffic is exchanged in plaintext; vulnerable to **spoofing, replay, and eavesdropping**.
- **Networking:** VPN trust is "flat"; any peer on the network can access all services without application-layer separation.
- **Data Storage:** Logs, rosbags, and telemetry (Grafana/Prometheus) are unencrypted, risking exposure of sensitive operational data.
- **Quantum Risk:** No PQC-resistant authentication or robot attestation exists; current classical identity management is vulnerable to future quantum threats.

6.5 Components and Roles Considered

This section defines the core entities and logical components that interact within the AMR fleet ecosystem. These components form the foundation upon which the PiQASO security functionalities will be integrated.

System Entities and Roles

- **Fleet Manager (Open-RMF host):** The central orchestrator of the system. It is responsible for task negotiation, traffic management, and fleet-wide coordination. It runs on a high-performance central server (Intel i7, 16GB RAM, Ubuntu 22.04) and acts as the primary authority for task assignments.
- **Autonomous Mobile Robots (TIAGo OMNI Base):** Each AMR is equipped with an on-board PC (Intel i5/i7, 8–16GB RAM, Wi-Fi + Ethernet) and runs the complete ROS 2 Navigation2 stack for local perception, path planning, and obstacle avoidance.

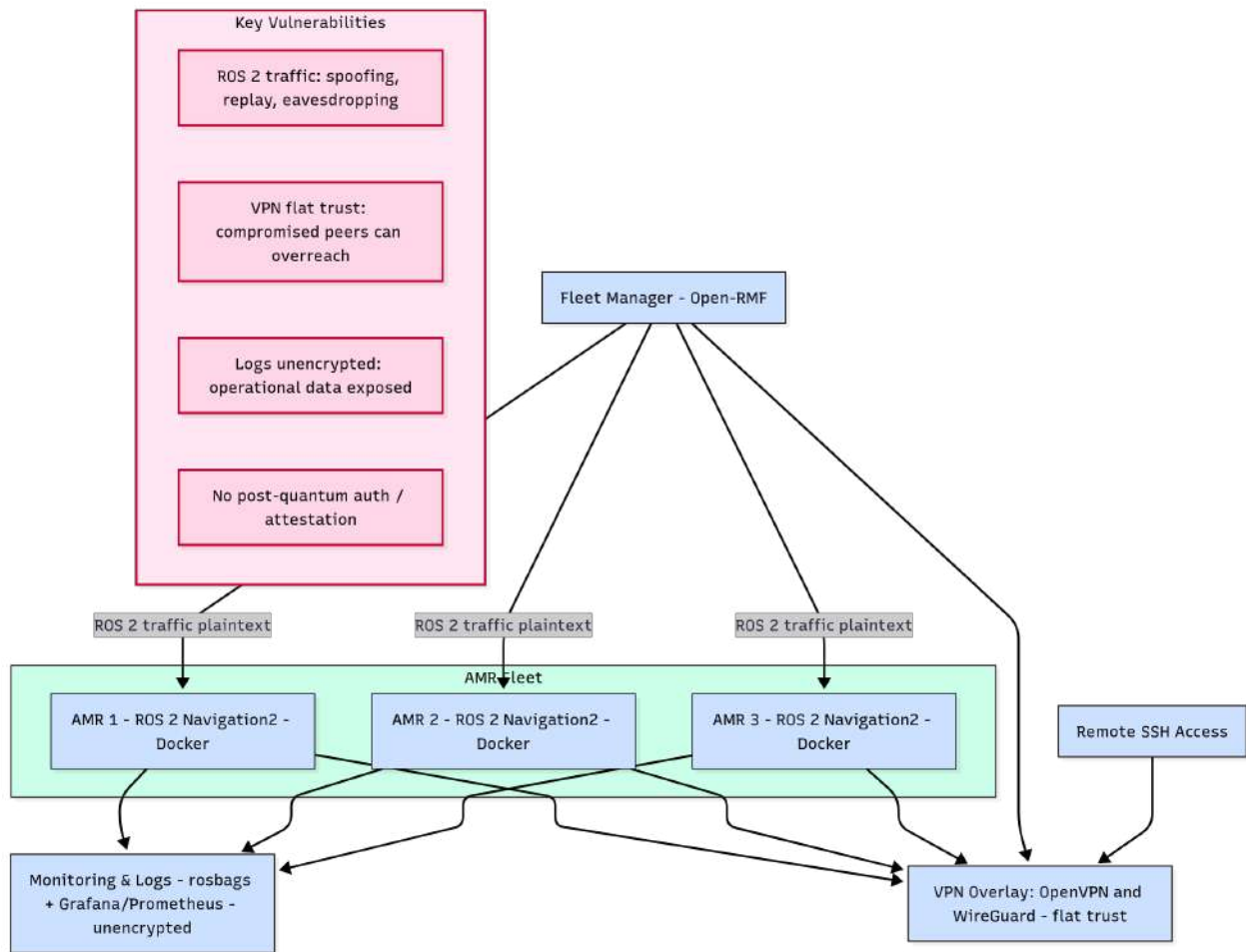


Figure 6.1: As-is AMR fleet communication architecture (VPN overlay with unprotected ROS 2 traffic).

- **Operators and End Users:** Human actors responsible for high-level supervision. They interact with the system to submit delivery requests and monitor fleet performance and health through visualization dashboards.

Communication & Middleware

- **ROS 2 / DDS Middleware:** The backbone of robot-to-robot and robot-to-manager communication. It handles the pub/sub mechanisms for all data topics. In the context of PiQASO, this component is the target for PQC-protected DDS plugins to ensure secure data exchange.
- **Exchanged Data Messages:**
 - **Navigation & Safety Data:** High-frequency messages containing robot poses, velocity commands (Twist), and obstacle detections from Lidar/Camera sensors.
 - **Mission & Status Data:** Lower-frequency messages used for task allocation, battery status updates, and mission completion confirmations.
 - **Operational Logs:** Data recorded during missions used for later debugging, performance analysis, and system monitoring.

Networking

- **Network Infrastructure:** Provides the physical and logical connectivity through Wi-Fi 802.11ac and site-level VPN overlays (OpenVPN/WireGuard).

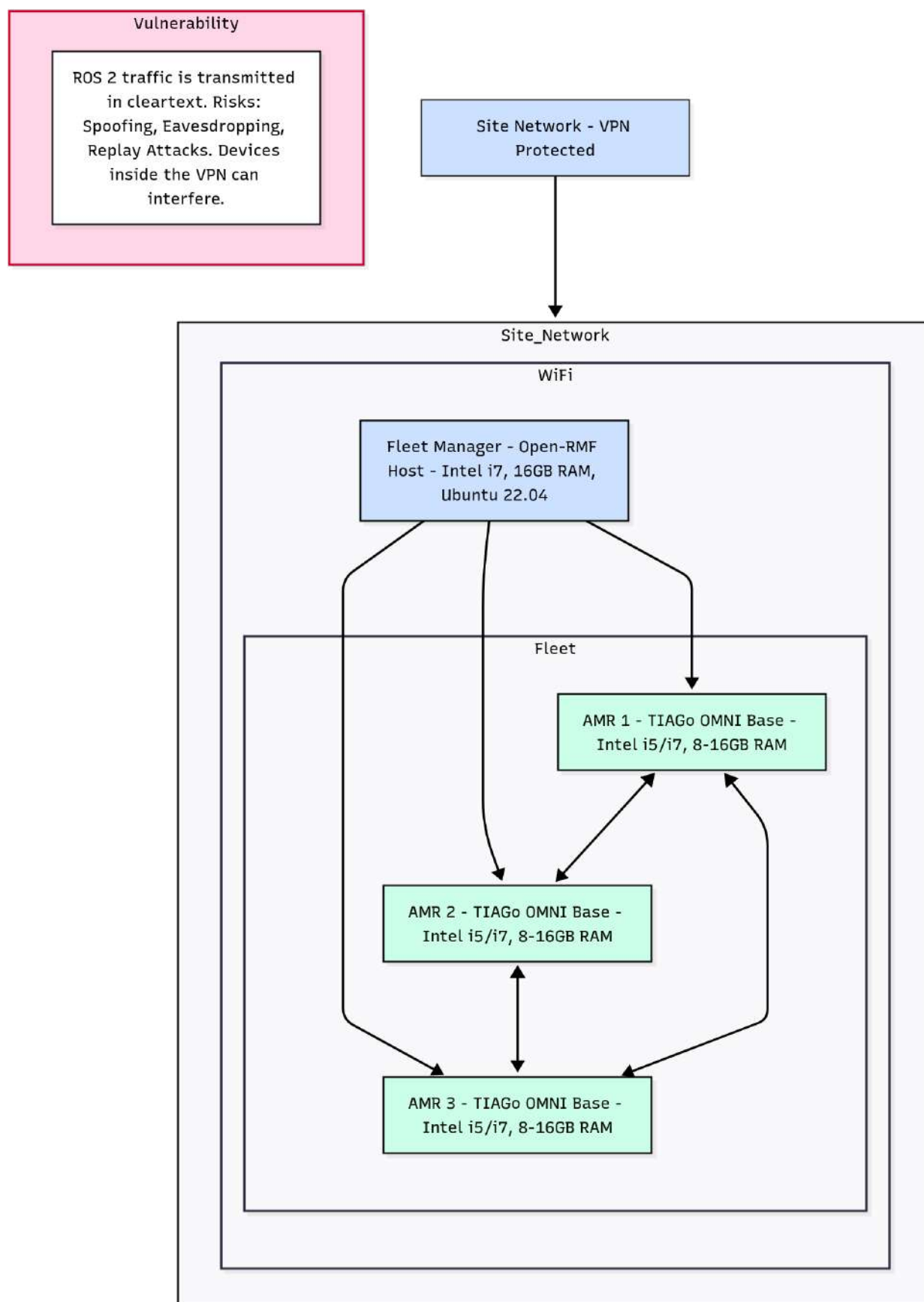


Figure 6.2: Logical component diagram of the Fleet Manager, AMRs, middleware, and operators.

6.6 Challenges and Needs from PiQASO

The Automation Use Case brings forward specific challenges that PiQASO must address in order to ensure quantum-safe and operationally viable fleet communications:

- **Strong Identity & Onboarding:** Each AMR and backend component must be uniquely identified and authenticated before joining the fleet. Current approaches rely on static certificates and VPN keys, which are insufficient against future quantum adversaries. A scalable, secure onboarding mechanism is required.
- **Fine-Grained Authorization:** Today, once a robot or operator gains network access, it can potentially reach all services. Instead, permissions should be differentiated: for example, basic telemetry access for operators, but only privileged users or systems can issue critical commands such as emergency stops or reroutes.
- **Confidentiality & Integrity of Communications:** Messages exchanged within ROS 2 and OpenRMF (including navigation data, task assignments, and status updates) must remain confidential and tamper-proof. This requires communication protection mechanisms that are suitable for real-time robotic workloads.
- **Selective Data Sharing:** Not all data should be shared with all robots or users. Task assignments, for instance, must only be readable by the designated AMR, while fleet-wide broadcasts (such as pause commands) must be securely disseminated to all. Mechanisms that support such selective sharing are needed.
- **Secure Data-at-Rest:** Logs, maps, and recorded operational data (rosbags) represent valuable assets both for debugging and for AI training. They must be stored in a way that protects confidentiality, prevents tampering, and ensures that only authorized entities can access them.
- **Maintaining Real-Time Performance:** Security mechanisms must not compromise the responsiveness of the fleet. Encryption, authentication, and policy checks need to be lightweight enough to preserve deadlines in navigation and task execution loops. Benchmarks and integration guidelines are required to ensure this balance.

6.7 “To-Be” Reference Scenario #1: Secured ROS 2 Communications

Purpose within PiQASO: This scenario addresses the core need to protect real-time navigation and control data. It demonstrates the transition from a flat network trust model to an application-layer model where every ROS 2 message is encrypted and signed. This ensures that even if the VPN is breached, the robot’s kinetic movements remain under authorized control.

PiQASO Components Required:

- **PiQASO SDK:** For end-to-end encryption of ROS 2 topics.
- **PQ-TLS (ML-KEM):** For establishing secure handshakes between AMRs and the Fleet Manager.
- **PQ Signatures (ML-DSA):** For authenticating mission-critical commands.

- **QR-TCB (FPGA):** For offloading cryptographic primitives to maintain real-time performance (Phase 2).

System Architecture Integration: The PiQASO SDK is integrated directly into the ROS 2 / DDS middleware layer of the TIAGo OMNI Base robots, acting as the application-layer security boundary for all high-frequency navigation and safety topics. For the initial deployment phase, the SDK handles all cryptographic logic in software. In the second version of the system, the FPGA-based QR-TCB will be interfaced with the robot's on-board PC to offload these primitives, enabling hardware-accelerated signature verification in microseconds. On the orchestration side, the Fleet Manager leverages the cloud-based PQCaaS backend for identity management, establishing a secure PQ-TLS communication backbone.

Expected Benefits

- Prevents unauthorized injection of malicious control data through hardware-rooted trust (**QR-TCB**) in the final version.
- Ensures session integrity against quantum adversaries using standardized **ML-KEM** handshakes.
- **Reduced Attack Surface:** Ensures secure session termination immediately after data exchange.

6.8 “To-Be” Reference Scenario #2: Secured Fleet Manager with Selective Sharing and Protected Storage

Purpose within PiQASO: This scenario addresses the need for granular access control for long-term storage and secure on-demand data exchange. It demonstrates two distinct cryptographic workflows: **Attribute-Based Encryption (ABE)** for role-based archival of sensitive AMR data (e.g., facility maps, rosbags), and **Updatable Public Key Encryption (UPKE)** for secure, on-demand sharing with external authorized entities.

PiQASO Components Required for Instantiation:

- **Attribute-Based Encryption (ABE) Module:** Enables policy-driven encryption where decryption is tied to specific user attributes (e.g., "Role: Admin").
- **Updatable Public Key Encryption (UPKE):** Facilitates secure on-demand sharing by allowing the sender to encrypt data specifically for a verified receiver upon request.
- **PQCaaS Backend (KMS):** Acts as the trusted authority that manages encryption policies for archival and orchestrates the sharing requests between receivers and senders.

System Architecture Integration: For archival, the **ABE module** is integrated into the AMR logging stack; the robot encrypts data using a policy-based public key provided by the PQCaaS before storage. For sharing, the system utilizes the **UPKE protocol**: when a receiver requests access, the PQCaaS coordinates with the AMR to generate a receiver-specific encrypted payload. This ensures that commercially sensitive industrial layouts and logs are only exposed to authorized entities under specific, audited conditions.

Expected Benefits

- **Incremental Security:** Protection of proprietary maps against future quantum decryption through **UPKE**-based forward secrecy.
- **Role-Based Privacy:** **ABE** ensures that data access is tied to identity attributes, preventing a single compromised account from accessing the entire data history.
- **Long-Term Confidentiality:** Hardware acceleration allows the use of complex **ABE** algorithms without introducing bottlenecks in data archiving.

6.9 “To-Be” Reference Scenario #3: ASSIST – Secure Remote Assistance

Purpose within PiQASO: This scenario provides a quantum-safe “emergency tunnel” for human intervention in autonomous operations. It addresses the critical vulnerability of remote assistance links, ensuring that manual control of an **AMR** cannot be hijacked by quantum-capable adversaries. To meet the ultra-low latency requirements of manual teleoperation, the system utilizes a **KEM/PDK-based TLS variant**. This approach bypasses traditional ephemeral handshakes by using pre-established public keys and signature-based authentication.

PiQASO Components Required for Instantiation:

- **PQ-TLS (KEM/PDK Variant):** Optimized handshake using pre-distributed keys to minimize connection setup time.
- **One-Time Signatures (WOTS):** Used for per-message authentication of velocity commands. WOTS offers faster signature generation on the controller side compared to ML-DSA.
- **QR-TCB (FPGA):** Performs high-speed signature-based verification to ensure manual commands are processed in < 10ms.
- **Attribute-Based Encryption (ABE):** Used exclusively to seal session logs for authorized auditing (UPKE is excluded from this scenario).

System Architecture Integration: The **ASSIST** tunnel is established after pre-loading established certificates onto the Edge Device. The handshake is offloaded to the **QR-TCB** via signature-based authentication. During the session, the Remote Technician station signs each velocity command using **WOTS**. The robot’s QR-TCB verifies these signatures, balancing the trade-off between ultra-fast generation (technician side) and hardware-accelerated verification (robot side). Finally, the entire session log is sealed via **ABE** and stored.

Expected Benefits

- **Minimal Handshake Latency:** PDK eliminates the need for complex key exchanges during emergencies.
- **Real-Time Command Integrity:** WOTS allows the technician to sign high-frequency command streams without CPU bottlenecks.
- **Secure Auditing:** ABE ensures that session logs are only accessible to certified safety investigators.

6.10 Reference Scenario User Stories and Metrics of Success

6.10.1 User Story #1 (Scenario #1): Secured Task Assignment

Planned for the first reporting period, this scenario focuses on securing real-time ROS 2 traffic. While Figure 6.3 illustrates the complete vision including hardware acceleration, the **first milestone focuses on a software-based implementation via the PiQASO SDK**. The transition to hardware-based FPGA acceleration (QR-TCB) is planned for the second version of the system. The Zeno framework manages the transport, while the primary focus remains the cryptographic protection of the message payload.

UC2.US1: Secure Mission-Critical Command Assignment to AMR

As a Fleet Manager, I want to assign a mission-critical command to an Autonomous Mobile Robot (AMR) through a channel that is quantum-resistant, verifiably authenticated as originating from me, and securely terminated immediately after the transaction so that the navigation data remains unreadable and untamperable, while minimizing the attack surface exposed during the session.

User Story Workflow

1. **Command Initiation:** The Manager initiates a mission command (e.g., "Navigate to Loading Dock 1").
2. **Secure Channel Establishment:** The SDK establishes a **PQ-TLS session**. For the current version, the **ML-KEM** handshake is performed in software by the SDK (simulating the QR-TCB role).
3. **Message Signing:** The command payload is signed using **ML-DSA (Dilithium)** within the SDK to ensure non-repudiation.
4. **Transmission:** The encrypted and signed payload is delivered via the Zeno-based PQTLS tunnel.
5. **Software-based Verification:** Upon receipt, the AMR's SDK decrypts the channel and verifies the digital signature using its software-based cryptographic library.
6. **Execution:** Once the integrity and origin are confirmed, the robot executes the task.
7. **Secure Session Termination:** After execution and acknowledgment, the SDK **smoothly terminates the PQ-TLS session**, clearing ephemeral keys from memory to reduce the window of opportunity for potential exploits.

The VPN overlay remains in place for additional network isolation, but the core trust is shifted to the application layer. The main difference from the "As-is" system is that security is no longer dependent on the assumption that all peers on the VPN are trustworthy. By enforcing a "connect-execute-terminate" lifecycle, the system ensures that long-lived idle sessions do not remain open, significantly hardening the fleet's security posture against unauthorized access.

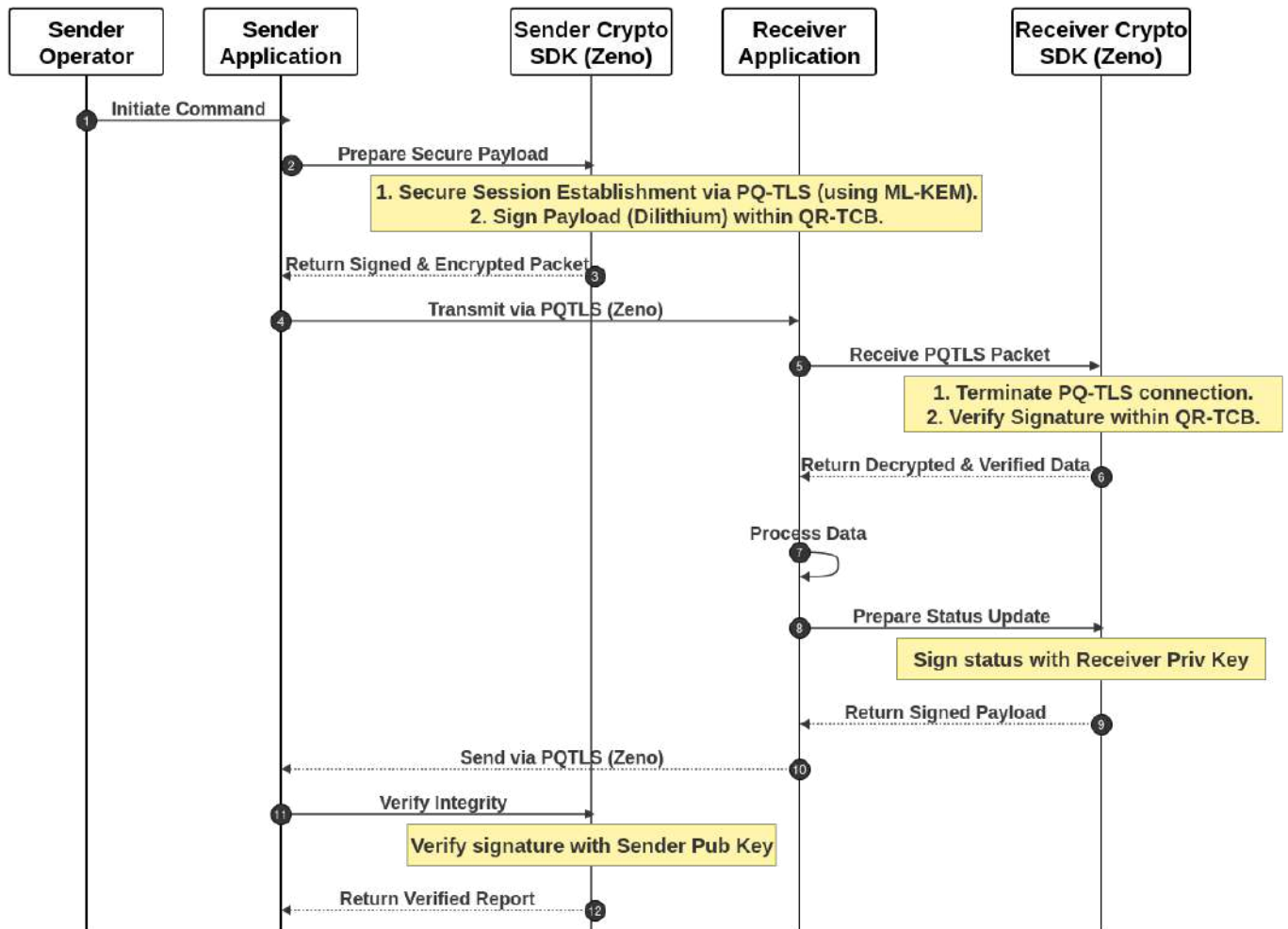


Figure 6.3: To-be Scenario #1: Secured ROS 2/DDS communication for navigation and task allocation.

6.10.2 User Story #2 (Scenario #2): Protected Data Logging with Evolutionary Security

UC2.US2: Policy-Based Sealing and Secure On-Demand Sharing of Rosbags

As a System Integrator / Security Admin, I want to ensure that generated rosbags are cryptographically sealed using policy-based access control (Attribute-Based Encryption) and can be securely shared on-demand with authorized receivers via key updates (UPKE) so that access to recorded robot data is restricted to entities meeting the required attributes, or sharing permissions can be updated over time without compromising the confidentiality of past or future data.

To ensure long-term confidentiality and flexible access, the system distinguishes between archival and active sharing. **ABE** is used for the former, trusting the PQCaaS to enforce storage policies, while **UPKE** is used for the latter to provide end-to-end security between sender and receiver.

- **Secure Archival via ABE (Policy-Driven):** This mechanism introduces fine-grained authorization for data-at-rest. Encryption is driven by specific policies (e.g., "Role: Admin" or "Role: Operator"). This ensures selective decryption: for instance, an operator can only decrypt telemetry data, while an administrator can access full operational rosbags. The **FPGA-based hardware acceleration** is

utilized to handle the high computational load of ABE primitives, ensuring that edge performance remains unaffected during log sealing.

- **Secure Sharing via UPKE (On-Demand):** Unlike archival, this mechanism is triggered only when a verified receiver requests access to data. Following the UPKE protocol, the AMR (sender) re-encrypts the requested data specifically for the receiver. This unidirectional process ensures that data is protected during transit and shared only with the intended party, providing end-to-end security without storing archival data under UPKE.

Flow A: Secure Archival (Attribute-Based Encryption)

1. **Request Policy:** The Edge Device (AMR) sends a request for an encryption policy to the PQCaaS.
2. **Policy Key Issuance:** The PQCaaS returns a **Policy-Based Public Key (ABE)**. By trusting the PQCaaS, the AMR relies on the service to enforce the attribute-based storage policy (e.g., "Role: Admin").
3. **Encryption & Storage:** The SDK encrypts the data locally using the ABE key (FPGA accelerated) and uploads the sealed archive to the storage.

Flow B: Secure On-Demand Sharing (UPKE)

1. **Access Request:** A specific Receiver (e.g., an Auditor) requests access to centrally stored archival data via the PQCaaS.
2. **Sender Notification:** The PQCaaS notifies the original Sender (the AMR) about the request.
3. **Specific Re-encryption:** The AMR generates a **new public key**, encrypts the requested data specifically for that receiver, and sends it back to the PQCaaS.
4. **Delivery:** The PQCaaS forwards the re-encrypted data to the Receiver for final decryption.

6.10.3 User Story #3 (Scenario #3): Secure Remote Intervention

UC2.US3: Secure Manual Takeover of a Stalled Robot

As a Remote Technician / Fleet Operator, I want to securely take manual control of a stalled robot through an interaction that is authenticated, encrypted, and hardware-verified so that the robot's kinetic functions cannot be hijacked by unauthorized parties during the manual takeover.

When a TIAGo OMNI Base robot identifies a situation it cannot resolve autonomously (e.g., a blocked path that requires a complex maneuver or a sensor anomaly), it triggers an ASSIST request. A certified remote technician initiates the intervention process through the PiQASO SDK. To ensure the highest level of security, the remote intervention is treated as a high-priority task assignment issued by the Fleet Manager on behalf of the technician, thus utilizing the PQ Signatures functionality listed in the PAL capability table.

The connection is secured by an ephemeral session established using AKE (Authenticated Key Exchange) within a dedicated PQTLS tunnel, providing forward secrecy for the duration of the intervention. Teleoperation commands and sensor feedback are treated as critical mission payloads, protected via PQ KEM encryption. To maintain real-time responsiveness, every override command is verified in microseconds by the robot's FPGA-based QR-TCB. Finally, the session data is encrypted at the source using UPKE and stored in a rosbag sealed with Attribute-Based Encryption (ABE) for authorized auditing.

User Story Workflow

1. **Deadlock Detection:** The AMR sends a PQC-signed assistance request to the Fleet Manager.
2. **Technician Authentication:** The Operator logs in using PQC-based credentials verified by the PiQASO SDK.
3. **Establish Secure Link:** An AKE-based ephemeral session is established inside a PQTLS tunnel, offloading the handshake to the QR-TCB (FPGA).
4. **Authenticated Override:** The Operator sends manual velocity commands; each packet is verified for integrity and authenticity by the robot's local FPGA.
5. **Session Archival:** Upon completion, the intervention data is cryptographically sealed using UPKE and ABE for future safety analysis.

6.10.4 Metrics of Success (KPIs)

To assess the success of UC2, measurable KPIs will be defined and benchmarked during integration and trials. These KPIs cover security, performance, scalability, and usability:

Table 6.2: KPIs for Use Case 2 (Automation)

KPI	Description	Acceptance Criteria
Security		
Unauthorized Task Rejection	Measure the system's ability to identify and reject mission commands from unauthenticated sources.	100% of unauthorized task assignments are rejected.
Fleet Access Control	Ensure that only robots with valid PQC-based identities can join the operational fleet.	100% of unauthorized robots prevented from joining the fleet.
Log Integrity	Detection of any attempt to modify or delete operational logs and rosbags.	Tampering attempts on logs are always detected.
Performance		
Task Assignment Latency	Impact of PQC signatures and PQ-TLS handshakes on the end-to-end command flow.	Total end-to-end target: ≤ 800 ms (≤ 300 ms for safety-critical functions).
Navigation Loop Latency	Measure the round-trip time for navigation data (poses, velocity) under PQC protection.	Total process time between 50 and 200 ms .
Compute Overhead	Onboard CPU resources consumed by PQC operations vs. standard navigation tasks.	SDK $\leq 10\%$; QR-TCB $\leq 35\%$ CPU overhead.
Scalability		
Secure Storage	Efficiency of PQC-protected storage and retrieval for large operational datasets.	Access delay for MB-range datasets ≤ 5 seconds .
Usability		
Role-based Policies	Correctness of Attribute-Based Encryption (ABE) in differentiating access between roles.	Role-based policies enforced correctly in 100% of tested operator/admin interactions.
Sealed Log Access	Ease for authorized system integrators to access and decrypt protected logs.	Authorized access with average delay ≤ 5 seconds .

These metrics ensure that security is not only cryptographically sound but also operationally viable in robotic environments where timing and responsiveness are critical.

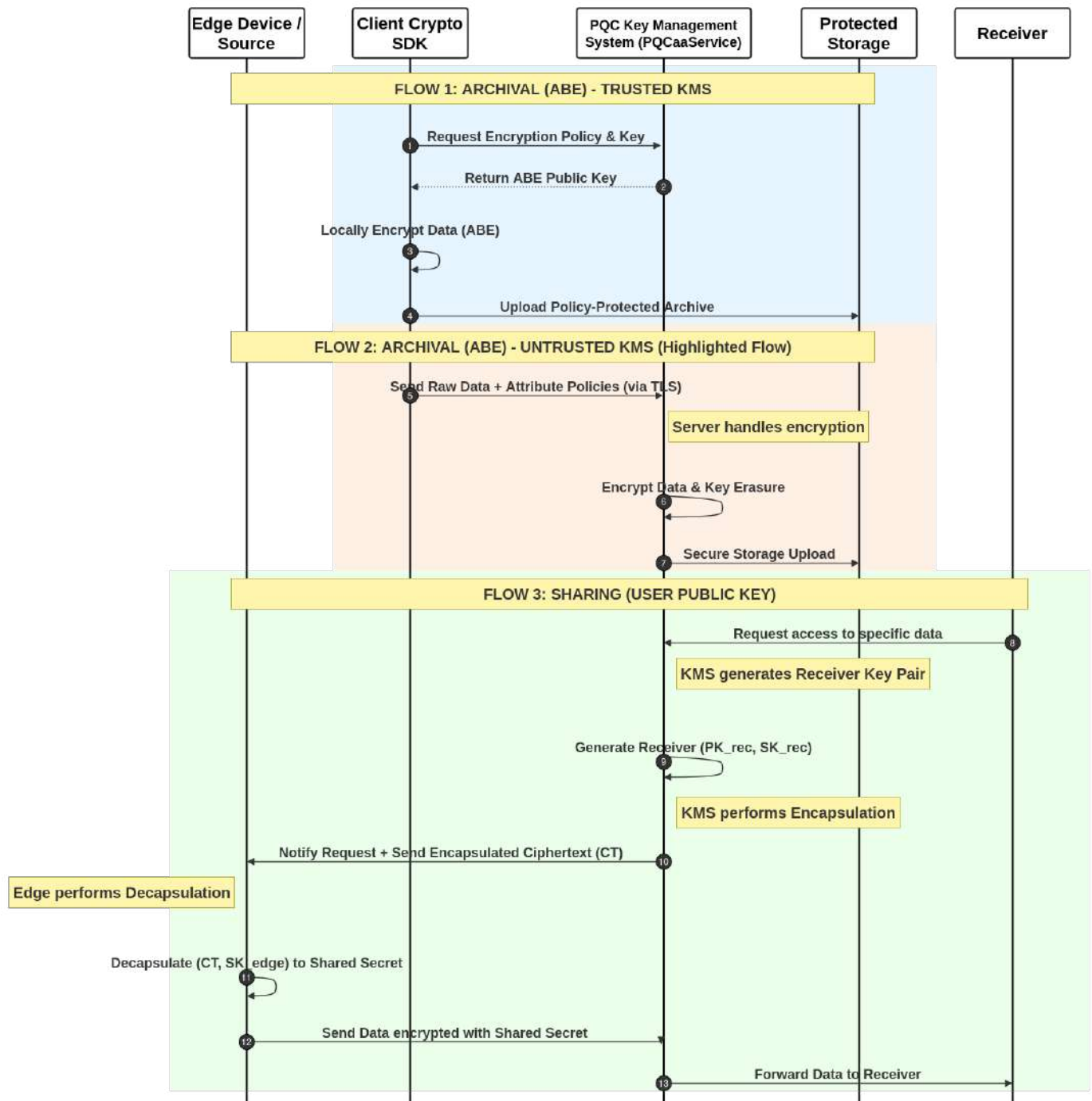


Figure 6.4: To-be Scenario #2: Secured Fleet Manager with selective sharing and protected storage.

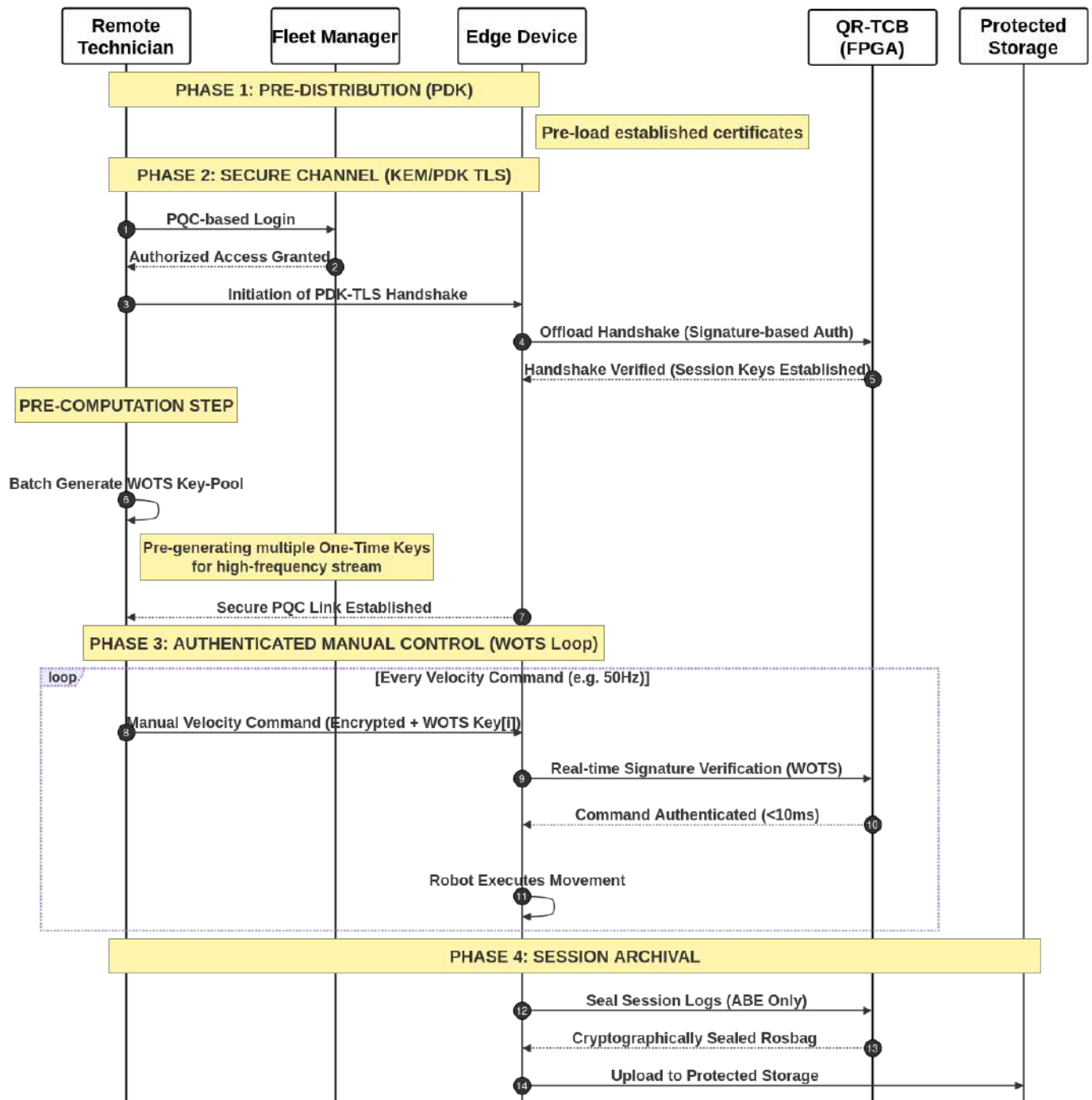


Figure 6.5: To-be Scenario #3: ASSIST – Secure Remote Assistance with AKE-based communications and hardware-verified manual control.

Chapter 7

PiQASO UC3 – Sector: Finance:

7.1 Use case's Description, Goals and Objectives

The third Use Case of PiQASO concerns the fortification of online financial transactions against emerging quantum technologies, which pose a critical threat to e-banking operations and the whole financial sector external communications.

The financial services industry has long been a lucrative domain for malicious parties attempting to compromise financial transactions and obtain illicit access to funds. As quantum computers have the potential to break widely used cryptographic algorithms, the need for PQC becomes increasingly urgent to safeguard online financial transactions against future quantum attacks. The implications of such attacks on the security and integrity of the aforementioned transactions are profound, posing significant risks to sensitive financial data, customer privacy, and overall trust in the financial system. In this regard, the adoption of PQC in the financial domain poses unique challenges, as there is a need to not only **enable transactions only between authenticated users**, but also **achieve forward secrecy in data management for ensuring that past transactions remain secure even if the key is compromised**, while also **avoiding the disclosure of sensitive infrastructure details**.

Although PQC holds promise for securing sensitive data against the threat of quantum attacks, its adoption in real-world systems poses several challenges and considerations for stakeholders in the online payments sector. These challenges span technical, operational, and regulatory domains (compatibility with existing infrastructure, performance considerations, key management, interoperability and standardization, etc.) and must be carefully addressed to ensure the successful integration of PQC solutions into existing infrastructure. By proactively addressing these challenges, the financial sector can strengthen their cybersecurity posture, mitigate the risks posed by quantum computing, and maintain trust and confidence in the integrity of transactions and data.

In this scenario, we want to simulate a customer initiating a online payment or e-banking operation. These requests traverse the CXB infrastructure before leaving back to the customer's endpoint. However, some of the current CXB's hardware are not prepared to integrate PQC. This creates a problem: Even if internal systems adopt PQC, the last mile to the outside world may still be vulnerable if the final piece of hardware cannot support PQC algorithms.

To address this limitation, PiQASO introduces a cloud-based PQC-as-a-Service model, where the back-end securely handles the heavy PQC operations (encryption, decryption, key exchange, digital signatures). In this approach, CXB can implement PQC regardless of hardware capabilities, ensuring that external communications with customer devices are protected against quantum adversaries. The "outdated" hardware remains in place, but PQC operations are offloaded to the service layer behind it, thereby creating a quantum-secure perimeter without requiring hardware replacement.

The primary focus of this Use Case is set on external communications (customer-to-bank), which are the most exposed and critical for customer protection and trust. Only once these channels are quantum-safe will CXB gradually extend PQC protections to internal communications within its infrastructure. Internal flows remain a secondary priority, as they already benefit from strong protections and controlled environments.

7.2 PiQASO Functionalities

Table 7.1: PiQASO Functionalities to be evaluated for Use Case 3 (Finance)

PiQASO Functionality	Description and Purpose in the Use Case	RP1	RP2
PQ-TLS	Secures the client-to-edge and edge-to-gateway session using TLS 1.3 with PQC or hybrid mechanisms, with PiQASO translating the protected session into classical TLS towards internal CXB systems that cannot natively support PQC.	✓	
PQ Signatures	Used together with key encapsulation mechanisms as part of the PQC operations (encryption, decryption, key exchange, digital signatures) offloaded to the backend, ensuring authenticity of client-bank transactions.	✓	
QR-TCB	Secure storage of cryptographic keys and supporting the execution of PiQASO PQC enablers.	✓	✓
PQCaaS	Allows the CXB infrastructure to consume PiQASO as a service, introducing PQC protections on top of existing communication channels without deploying dedicated on-premise infrastructure.		✓
Hardware Acceleration	Offloads computationally intensive PQC primitives (key encapsulation mechanisms, digital signatures) to an FPGA accelerator, either on-premise or cloud-based, to mitigate performance and latency impact on banking operations.		✓

7.3 “As-is” Scenario

An example of how online banking encryption works is provided in Figure 7.2. In this simple example, we can see how the CDN is responsible for the key distribution in the current CXB environment. In a post-quantum scenario, this poses a problem since Shor’s algorithm will be able to break this encryption.

7.4 Components and roles considered in the context of Secure Online Payments Use Case

In this section, we describe the components and roles involved in the Secure Online Payments Use Case within the PiQASO framework. These elements are critical to ensure secure, resilient, and post-quantum-ready communications between external clients and the internal banking infrastructure.

- **Routers UNIC (ASR-9901):** These routers serve as the entry points for external internet traffic, connecting the bank’s infrastructure to providers such as Telefónica and Verizon. They are the first layer of control and filtering before traffic enters the secure zones.

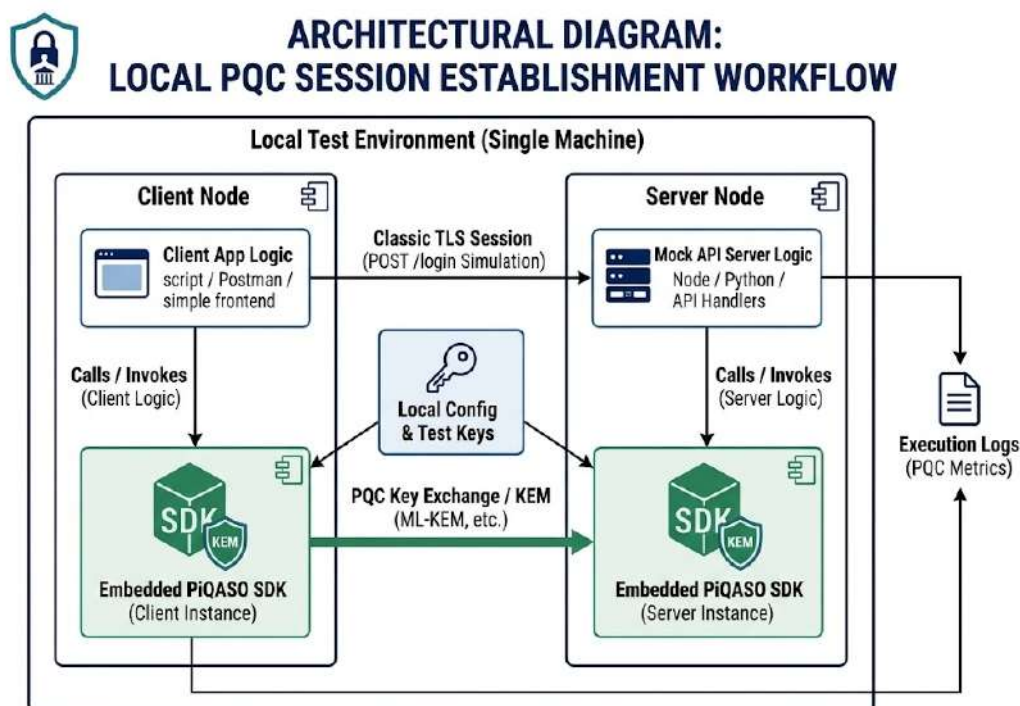


Figure 7.1: Financial e-banking architecture and workflow.

- **Switches UNIC (N9K-C93180YC-FX3):** These switches distribute incoming traffic across the network, ensuring efficient routing and segmentation between different zones such as DMZ, CD1/CD2, and backend systems.
- **External Firewalls GICAS (SRX-5400):** These firewalls provide perimeter security by filtering and inspecting incoming traffic. They enforce access control policies and protect against unauthorized access and external threats.
- **SSL Load Balancer (Thunder 5840S-11, upcoming upgrade to TH7655S):** The Secure Load Balancer distributes encrypted traffic across multiple front-end servers, ensuring availability, fault tolerance, and optimized performance for secure payment services. Today, this SLB SSL does not allow PQC, creating the problem we want to tackle with PIQASO, finding the balance between hardware restrictions and secure cryptography protocols.

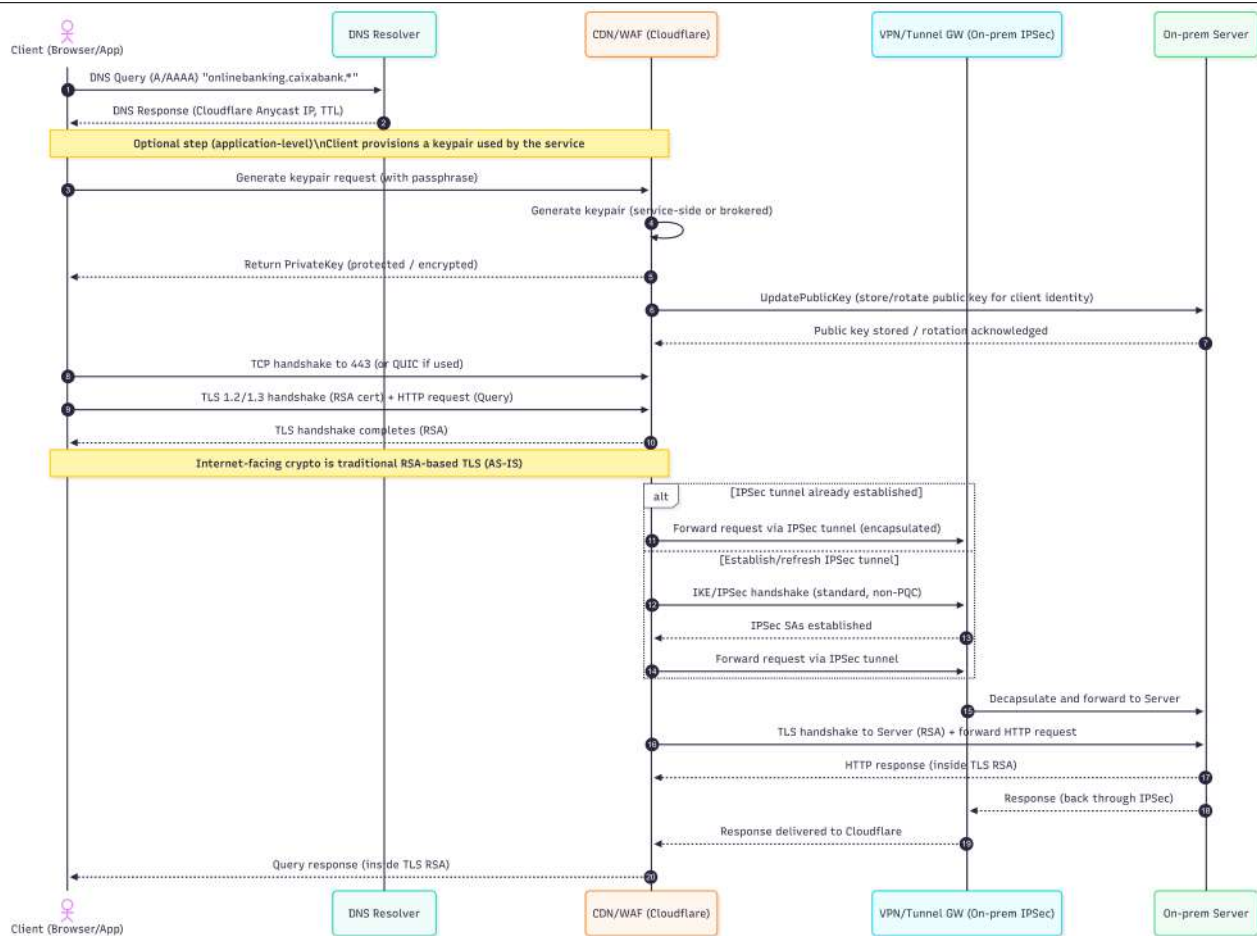


Figure 7.2: RSA Key distribution in actual online banking.

7.5 Challenges and Needs from PiQASO

The Secure Online Payments Use Case presents a unique set of challenges that PiQASO aims to address through its post-quantum cryptography (PQC) integration strategy. These challenges stem from the coexistence of legacy hardware, regulatory constraints, and the need for seamless customer experience in banking environments.

- **Hardware Limitations:** Several critical components in the current CXB infrastructure—such as the SSL Load Balancer (Thunder 5840S-11), external firewalls (SRX-5400), and routers/switches—do not natively support PQC algorithms. Replacing these devices is not feasible in the short term due to cost, operational risk, and vendor dependencies.
- **Last-Mile Vulnerability:** Even if internal systems adopt PQC, the final leg of communication with external clients remains vulnerable if the edge hardware cannot process PQC. This creates a security gap that undermines the integrity of the entire transaction flow.
- **Cryptographic Agility:** The infrastructure must support dynamic cryptographic transitions, allowing CXB to switch between classical and post-quantum algorithms based on context, client capabilities, and regulatory requirements. This agility is currently limited by static configurations in legacy systems.
- **Performance and Latency:** PQC algorithms are computationally intensive and may introduce latency in real-time payment flows. PiQASO must ensure that the integration of PQC does not de-

grade user experience or violate service-level agreements.

- **Key Management Complexity:** The introduction of new key types and exchange mechanisms requires a rethinking of key lifecycle management, including generation, distribution, rotation, and revocation. Existing systems are not designed to handle these complexities.
- **Regulatory Compliance:** Financial institutions operate under strict regulatory frameworks. Any cryptographic transition must be auditable, standards-compliant, and interoperable with third-party systems such as payment gateways and ile platforms.
- **Need for PQC-as-a-Service:** To overcome hardware limitations, PiQASO proposes a cloud-based PQC-as-a-Service model. This approach offloads cryptographic operations to a secure backend, enabling quantum-safe communications without requiring immediate hardware upgrades.
- **Trust Preservation:** The success of PQC integration depends on maintaining customer trust. Any disruption or failure in secure communications could have reputational consequences. PiQASO must ensure transparency, reliability, and continuity throughout the transition.

7.6 “To-Be” Reference Scenario: PQC Secured Online Banking Operations

Purpose within PiQASO: In the envisioned “To-Be” scenario, PiQASO enables secure online banking operations by integrating Post-Quantum Cryptography (PQC) into client-to-bank communication flows, while maintaining compatibility with existing RSA-based internal systems. This hybrid approach allows CXB to progressively introduce quantum-safe protections without requiring immediate changes to its current infrastructure.

This Use Case is structured around three complementary scenarios, each representing a different deployment and integration strategy of PQC within the banking architecture.

7.6.1 User Story #1 (Scenario #1): PQC Protection for External Communications (Edge Integration)

UC3.US1: PQC-Secured Edge Gateway for Client-Bank Communication

As a Bank Operator, I want to protect communications between the client and the bank by deploying PiQASO as a cryptographic gateway at the network edge, securing the client-to-edge and edge-to-gateway path with post-quantum cryptography (PQC) while maintaining interoperability with existing internal systems so that the most exposed segment of the communication channel is protected against quantum threats, while allowing gradual, non-disruptive adoption of PQC that preserves full compatibility with current operations.

In this scenario, PiQASO is deployed within the CXB infrastructure, acting as a cryptographic gateway at the network edge. Its role is to enable PQC-secured communication between the client and the bank, while ensuring seamless interoperability with existing internal systems.

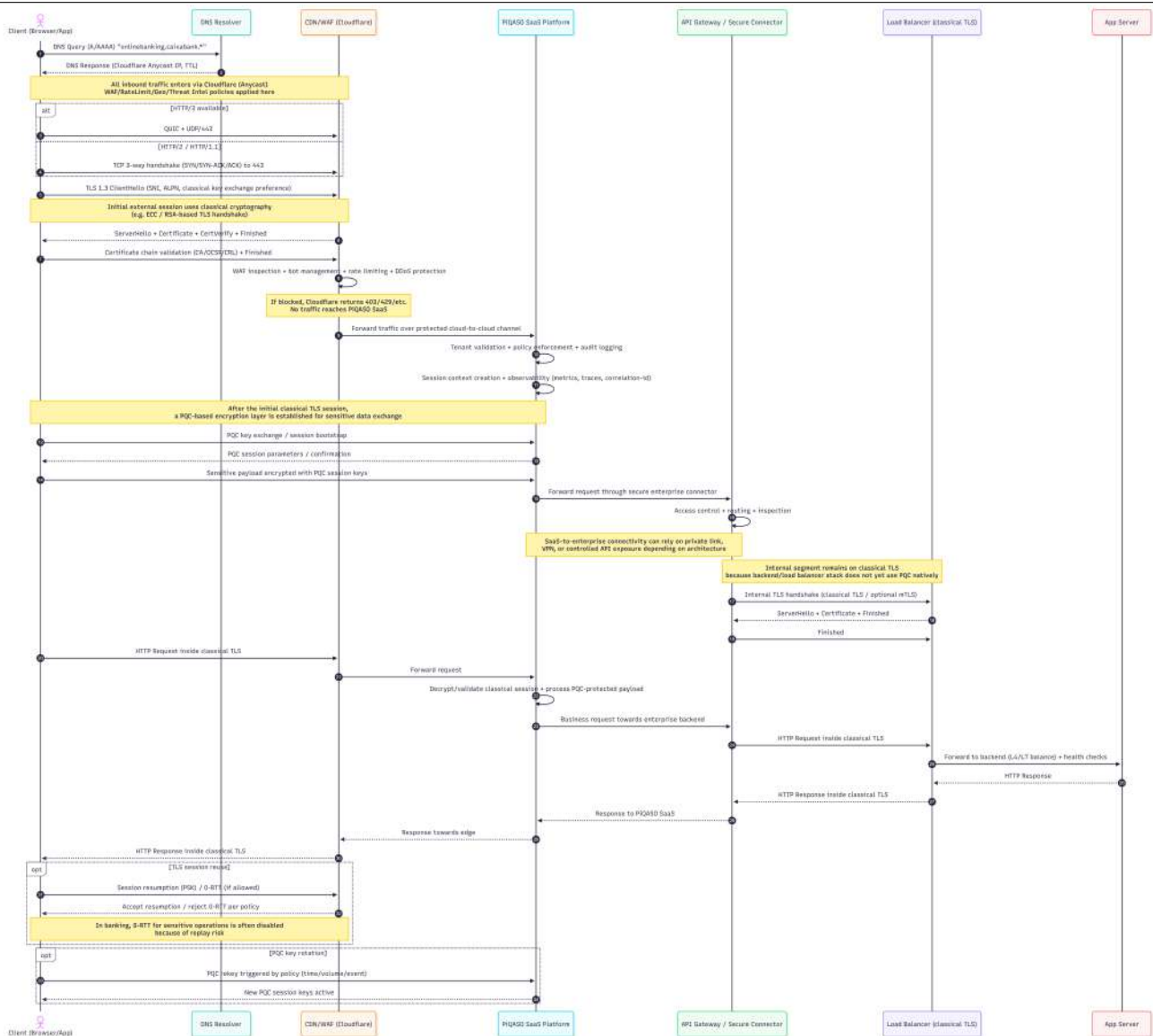


Figure 7.3: PQC Protection for External Communications via PiQASO.

The sequence of operations for this use case is illustrated in Figure 7.3 and can be described as follows:

1. The customer initiates a secure online banking session from their device. The request is established using TLS 1.3 with PQC or hybrid cryptographic mechanisms, ensuring quantum-resistant communication at the entry point.
2. The request is received at the edge (e.g., CDN/WAF), where standard security controls such as rate limiting, bot protection, and traffic inspection are applied.
3. The traffic is then forwarded through a secure channel to the CXB infrastructure, where PiQASO is deployed as an on-premise gateway.
4. Upon receiving the request, PiQASO performs cryptographic processing and enforces security policies. Due to the limitation of existing internal components (e.g., SSL Load Balancers), PiQASO translates the PQC-protected communication into a classical TLS session based on RSA or ECC.
5. The request is forwarded to the internal banking systems, where it is processed using the existing infrastructure without requiring PQC support.

6. The response follows the reverse path. PiQASO re-applies PQC protection before sending the response back to the client through the external edge.

This approach enables CXB to secure the most exposed segment of the communication channel—the client-to-edge and edge-to-gateway path—while maintaining full compatibility with existing internal systems. PiQASO acts as a cryptographic bridge, allowing the gradual adoption of PQC without disrupting current operations.

7.6.2 User Story #2 (Scenario #2): PQCaaS for External Communications (SaaS Model)

UC3.US2: PQC-as-a-Service Consumption

As a Bank Operator, I want to consume PiQASO as a service, introducing PQC protections on top of existing communication channels without deploying dedicated on-premise infrastructure so that PQC capabilities can be flexibly and rapidly adopted, externalizing cryptographic complexity while preserving control over sensitive operations and maintaining compatibility with internal systems.

In this scenario, PiQASO is consumed as a service, enabling CXB to leverage PQC capabilities without deploying dedicated on-premise infrastructure. PQC protections are introduced on top of existing communication channels, allowing for a flexible and scalable integration model.

The sequence of operations is illustrated in Figure 7.4:

1. The customer initiates a secure connection using standard TLS mechanisms with the CXB edge infrastructure.
2. The request is processed by the external security layer (e.g., CDN/WAF), which performs traffic filtering and protection functions.
3. The request is forwarded to the PiQASO SaaS platform through a secure channel. PiQASO applies policy enforcement and establishes a cryptographic context for the session.
4. A secondary PQC-based protection layer is established between the client and PiQASO, enabling the exchange of sensitive information with post-quantum guarantees.
5. PiQASO processes the request and forwards it to the CXB backend systems through a controlled and secure connector, typically using classical TLS compatible with existing infrastructure.
6. The response is processed by PiQASO, re-protected if required, and returned to the client following the same path.

This model provides flexibility and rapid adoption of PQC capabilities, allowing CXB to externalize cryptographic complexity while preserving control over sensitive operations and maintaining compatibility with internal systems.

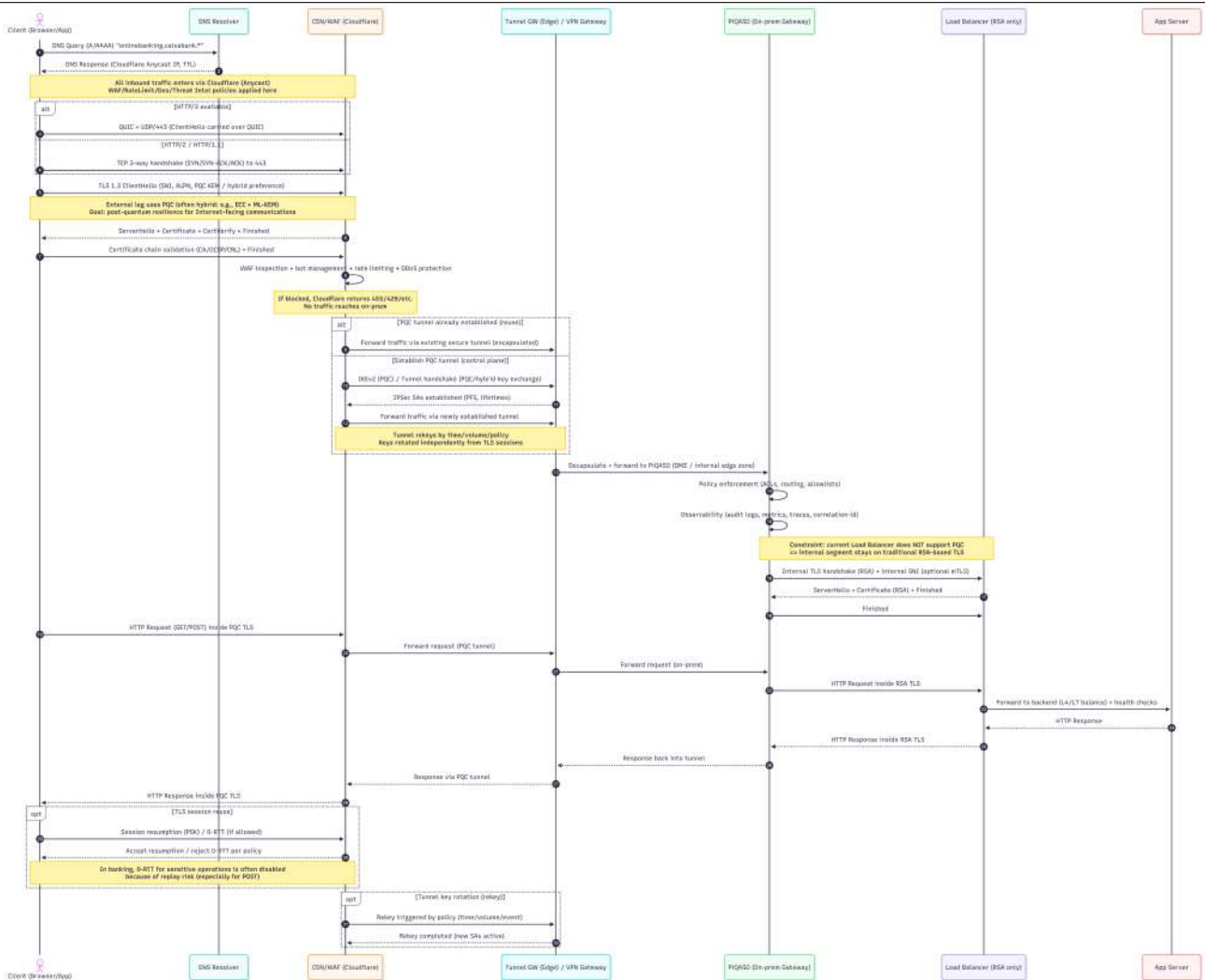


Figure 7.4: PiQASO SDK used as a PQaaS.

7.6.3 User Story #3 (Scenario #3): FPGA-Accelerated PQC Operations

UC3.US2: FPGA-Accelerated PQC

As a Bank Operator, I want to leverage PiQASO enhanced with hardware acceleration capabilities through FPGAs (Field Programmable Gate Arrays) so that the computational cost associated with PQC algorithms is addressed, enabling efficient, high-performance cryptographic operations without compromising security.

In this scenario, PiQASO is enhanced with hardware acceleration capabilities through the use of FPGAs (Field Programmable Gate Arrays) to address the computational cost associated with PQC algorithms.

The sequence of operations is illustrated in Figure 7.5 and extends the previous scenarios by introducing an optimized cryptographic processing layer:

1. The customer initiates a secure banking operation, which is forwarded through the external edge infrastructure following standard secure communication procedures.
2. The request reaches the CXB domain, where PiQASO identifies that the operation involves computationally intensive PQC primitives, such as key encapsulation mechanisms or digital signatures.

3. Instead of executing these operations purely in software, PiQASO offloads the cryptographic processing to an FPGA accelerator, either deployed on-premise or in a cloud-based environment.
4. The FPGA executes the PQC operations efficiently, generating the required cryptographic material (e.g., shared secrets or signatures) and returning the results to PiQASO.
5. PiQASO continues processing the request using the generated cryptographic outputs and forwards it to the backend systems through standard secure channels.
6. The response follows the reverse path, optionally leveraging FPGA acceleration again for cryptographic operations before being returned to the client.

By incorporating FPGA acceleration, CXB can mitigate the performance challenges associated with PQC adoption. This approach ensures that high-security cryptographic operations do not introduce unacceptable latency or impact user experience, making PQC deployment viable in performance-sensitive banking environments.

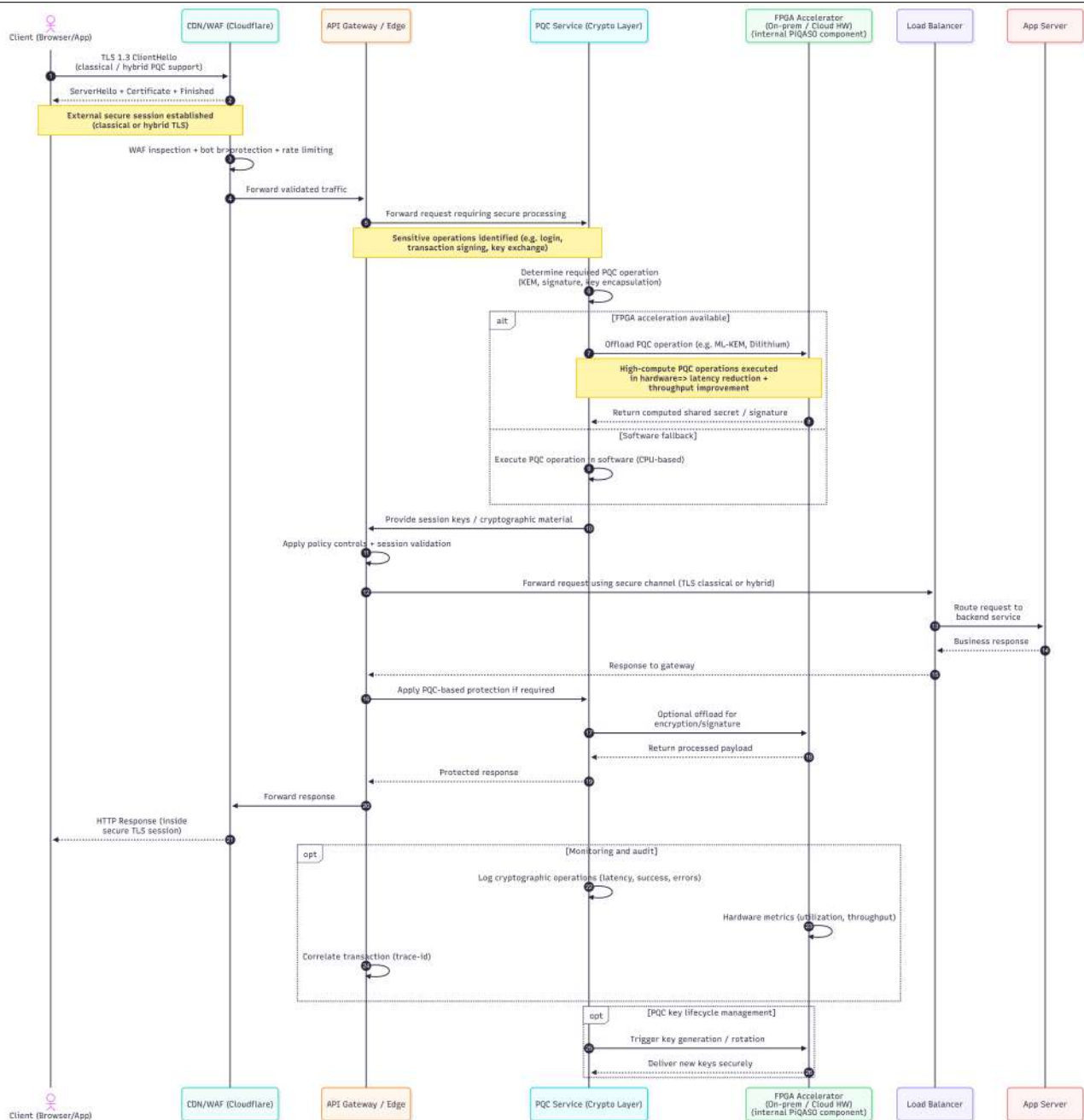


Figure 7.5: FPGA Hardware Acceleration Evaluation Testing.

7.6.4 Overall Architecture Perspective

Across all three scenarios, PiQASO enables a gradual and controlled transition toward quantum-safe communications. The combination of edge-based integration, service-based consumption, and hardware acceleration provides CXB with a flexible strategy to adopt PQC while preserving existing investments.

By decoupling cryptographic capabilities from infrastructure constraints, CXB can secure critical communication channels today while preparing for future quantum threats in a scalable and sustainable manner.

7.6.5 Metrics of Success (KPIs)

To assess the success of UC3, measurable KPIs will be defined and benchmarked during integration and trials. These KPIs cover security, performance, scalability, and usability:

Table 7.2: Consolidated UC3 KPIs for Security, Real-Time Performance, Scalability and Usability

KPI	Description	Acceptance Criteria
Security		
Command authenticity	Proportion of unauthorized or spoofed transaction requests rejected during authentication and signature verification	100% of unauthorized requests blocked
Sensor-tamper detection	Detection of tampering in transaction data (amount, beneficiary, metadata) during transmission	100% of tampered transactions detected and rejected
Real-Time Performance		
Closed-loop response (PQC)	End-to-end latency for transaction execution (client → gateway → backend → response) including PQC overhead	≤ 250 ms overhead vs baseline
Key-agreement latency	Time required to establish or rekey a PQC/hybrid secure session	$\leq 5\text{--}10$ s (aligned with session KPIs / p95 target)
Per-message crypto	Cryptographic processing cost per transaction message (encryption, signing, verification)	$\leq$ a few ms per message
SUT CPU overhead	Additional CPU load introduced on core banking/backend systems	Minimal / negligible impact (offloaded where applicable)
Scalability & Stress		
Node scalability	Number of concurrent users/sessions supported under PQC without performance degradation	$\geq$ baseline system capacity maintained
Secured throughput	Sustained throughput under PQC overhead compared to classical baseline	$\geq 90\%$ of baseline throughput
Stress headroom	Maximum load (transactions/sec, concurrent sessions) before SLA degradation	$\geq 2\times$ nominal load; failure threshold reported
Usability		
Setup time	Time required to configure and enable PQC in the application environment	≤ 5 minutes (configurable / toggle-based)
Workflow disruption	Impact of PQC integration on existing banking operations and user workflows	Zero disruption to existing workflows

Chapter 8

PiQASO UC4 – Sector: Energy

8.1 Use Case’s Description, Goals and Objectives

Use Case 4 (UC4) addresses the critical need to safeguard the energy sector’s digital infrastructure, particularly the electrical smart grid, against the looming threat of quantum-enabled cyberattacks. The energy sector operates through a highly interconnected and sensitive supply chain, involving green energy producers (MORE), smart grid operators (MOH), and monitoring and control specialists (CERth). These actors depend on seamless, secure, and real-time data communication for functions such as load balancing, grid status monitoring, control signal transmission, and energy distribution. The advent of quantum computing poses significant risks to the classical cryptographic methods currently in use, potentially compromising system integrity, reliability, and safety. UC4 leverages PiQASO’s Post-Quantum Cryptography-as-a-Service (PQaaS) framework to introduce robust cryptographic primitives, ensuring end-to-end data protection from the edge (energy production and monitoring) to the cloud-based control centres – while supporting both legacy systems and future-proof infrastructures.

From an application-specific perspective, the energy sector presents a uniquely demanding context for PQC adoption. The electrical smart grid operates in a safety-critical, real-time environment where cryptographic failures are not merely a data-security concern but a potential threat to physical infrastructure. Grid operators such as MOH must maintain uninterrupted telemetry between SCADA systems: any latency or availability impact caused by cryptographic transitions must remain within operationally acceptable bounds. At the same time, energy producers such as MORE operate large fleets of highly constrained IoT and OT field devices – photovoltaic inverters, energy meters, wind turbine sensors – that cannot host full PQC stacks locally and require a cloud-offload model (PQCaaS) to participate in a quantum-secure framework.

The “harvest now, decrypt later” (HNDL) threat is particularly acute in the energy sector: grid telemetry and production data collected today may retain long-term sensitivity (e.g. for infrastructure planning, competitive intelligence, or national security analysis), making it a prime target for adversarial HNDL campaigns already underway. This motivates the immediate need to begin PQC migration for communications that are encrypted today with RSA or ECC, before a cryptographically relevant quantum computer becomes available. Additionally, the long operational lifespan of smart grid assets (15–25 years for substations) means that any cryptographic architecture deployed now must support crypto-agility – the ability to transition between algorithm suites as standards evolve – without requiring hardware replacement. These factors together make UC4 a compelling and practically urgent use case for PiQASO.

The overarching goal of UC4 is to demonstrate how post-quantum cryptographic mechanisms can be pragmatically and effectively deployed in the context of critical energy infrastructure. Specifically, PiQASO aims to protect sensitive smart grid communications using hybrid cryptographic techniques, enabling

secure and authenticated data exchanges across heterogeneous systems without requiring substantial hardware modifications. The integration of the PiQASO SDK into smart grid nodes and endpoints, along with the use of a cloud-based PQC-as-a-Service backend, allows for the offloading of computation-intensive cryptographic tasks, ensuring security without performance trade-offs. Additionally, the solution introduces policy-based and attribute-aware encryption techniques that allow for fine-grained access control, supporting compliance, operational continuity, and regulatory mandates such as NIS2 and the Cyber Resilience Act. By demonstrating secure communication across the grid's lifecycle – from data generation to storage and retrieval – this use case contributes to the EU's strategic goal of quantum-resilient infrastructure.

UC4 sets out a multi-faceted set of objectives to operationalise PQC within the energy domain. First, it seeks to establish secure, quantum-resistant communication channels across the entire smart grid architecture by integrating PiQASO's lattice-based key encapsulation and digital signature schemes (e.g., ML-KEM/Kyber, ML-DSA/Dilithium) into the existing infrastructure. Second, it aims to provide device- and identity-level authentication and authorisation through a combination of pre-shared key techniques, PQ-TLS, and signature-based methods. Third, the scenario involves the deployment of attribute-based encryption (ABE) mechanisms that facilitate policy-driven access to encrypted data, ensuring that only authorised stakeholders can interact with specific data sets. Lastly, the use case includes the onboarding of constrained edge devices with secure bootstrapping methods and crypto-agile capabilities, preparing the sector for full-scale quantum readiness through secure lifecycle management and traceability.

8.2 PiQASO Functionalities

Table 8.1: PiQASO Functionalities to be evaluated for Use Case 4 (Energy)

PiQASO Functionality	Description and Purpose in the Use Case	RP1	RP2
PQ-TLS (incorporating ML-KEM and ML-DSA)	Establishes a mutually authenticated, quantum-resistant session between the gateway and the SCADA control centre, using ML-KEM for key encapsulation and ML-DSA for certificate-based mutual authentication of real-time grid telemetry.	✓	
PQ Signatures (ML-DSA)	Signs and verifies PQC certificates issued by the PiQASO PKI, enabling mutual authentication between gateway and SCADA and supporting certificate revocation checks.	✓	
QR-TCB	Secure storage of cryptographic keys and supporting the execution of PiQASO PQC enablers.	✓	✓
PQCaaS	Offloads certificate management, algorithm negotiation, and computationally intensive cryptographic operations for legacy SCADA components that cannot host the full PiQASO SDK, and later provides centralised key/credential delivery during algorithm transitions.	✓	✓
Crypto-Agility (multi-algorithm PQC library)	Enables a gateway to dynamically switch between supported PQC algorithm suites (e.g., ML-KEM-512 to ML-KEM-768, or co-establishing ML-KEM and HQC keys) without service interruption.		✓
ABE	Enforces policy-driven encryption of MORE's energy production data, ensuring that energy flow records and operational telemetry are decryptable only by authorised MOH roles and verified system components.		✓
PQ Signatures (FN-DSA/Falcon)	Used during zero-trust onboarding of edge gateways to sign the Certificate Signing Request and the issued X.509 identity token, establishing a compact, quantum-resistant root of trust.		✓

8.3 “As-Is” Scenario

In the current state, the energy sector’s smart grid relies entirely on classical cryptographic mechanisms across all communication layers, leaving the infrastructure exposed to quantum-enabled attacks. At the field edge, MORE operates three categories of constrained IoT devices , photovoltaic inverter sensors, energy meters, and wind turbine sensors , distributed across multiple production sites. These devices transmit real-time operational telemetry over local OT networks using industrial protocols such as Modbus to their respective on-site edge gateways. No cryptographic protection is applied at this level, as the protocols themselves provide none by default.

From there, the SCADA gathers telemetry data and forwards it over local-area network (LAN) channels to the gateway. These LAN links are protected using TLS 1.2, with RSA-2048 or ECC for key exchange and authentication. While adequate against classical adversaries, this represents the architecture’s most critical quantum vulnerability: both RSA and ECC key exchange are fully broken by Shor’s algorithm on a cryptographically relevant quantum computer, meaning that any adversary conducting “harvest now, decrypt later” collection on these links today will be able to decrypt the captured traffic in the future.

The gateway acts as the primary aggregation and relay point, consolidating telemetry and forwarding it upstream to the telemetry backend and frontend respectively, again over a classically protected TLS 1.2 channel using RSA or ECC. At this tier, the absence of any post-quantum-resistant authentication mechanism means that neither the gateway nor the SCADA system can verify each other’s identity in a quantum-resilient way. Moreover, the data platform , which hosts the database and the analytics services, over TLS 1.2 with classical PKI. Two further structural weaknesses are evident at this layer: there is no attribute-based or policy-aware access control governing which roles and system components may access specific data categories, and any analytics or computation on the stored data requires it to be fully decrypted first, exposing raw production and grid telemetry to any party with access to the processing environment.

Taken together, the as-is architecture has four distinct vulnerability classes that the PiQASO To-Be scenarios are designed to address: quantum-vulnerable key exchange on all LAN and SCADA links; the absence of post-quantum-resistant mutual authentication; the lack of attribute-based access control over shared energy data; and the inability to perform analytics on encrypted data without prior decryption.

The “As-Is” scenario highlights several structural weaknesses: the use of RSA and ECC-based authentication across SCADAs and gateways; the absence of quantum-resistant key exchange for SCADA/gateway communications; the lack of attribute-based or policy-aware access control for data shared between the grid operator and energy producers; and the inability to perform computations on encrypted data without first decrypting it. These shortcomings place both the integrity of critical grid operations and the confidentiality of commercially sensitive energy data at risk in the post-quantum era.

8.4 Components and Roles Considered

The following key actors, systems, and components are considered within the scope of UC4:

- **MOH - Smart Grid Operator:** Owner and operator of the smart grid infrastructure, including the central SCADA system, substation gateways, and the cloud-based data management platform. MOH is responsible for grid stability, load balancing, and real-time monitoring.
- **MORE - Green Energy Producer:** Operator of distributed renewable energy production sites (photovoltaic, wind). MORE manages constrained IoT and OT field devices (sensors, PLCs, edge gateways) that gather and transmit production telemetry to MOH.

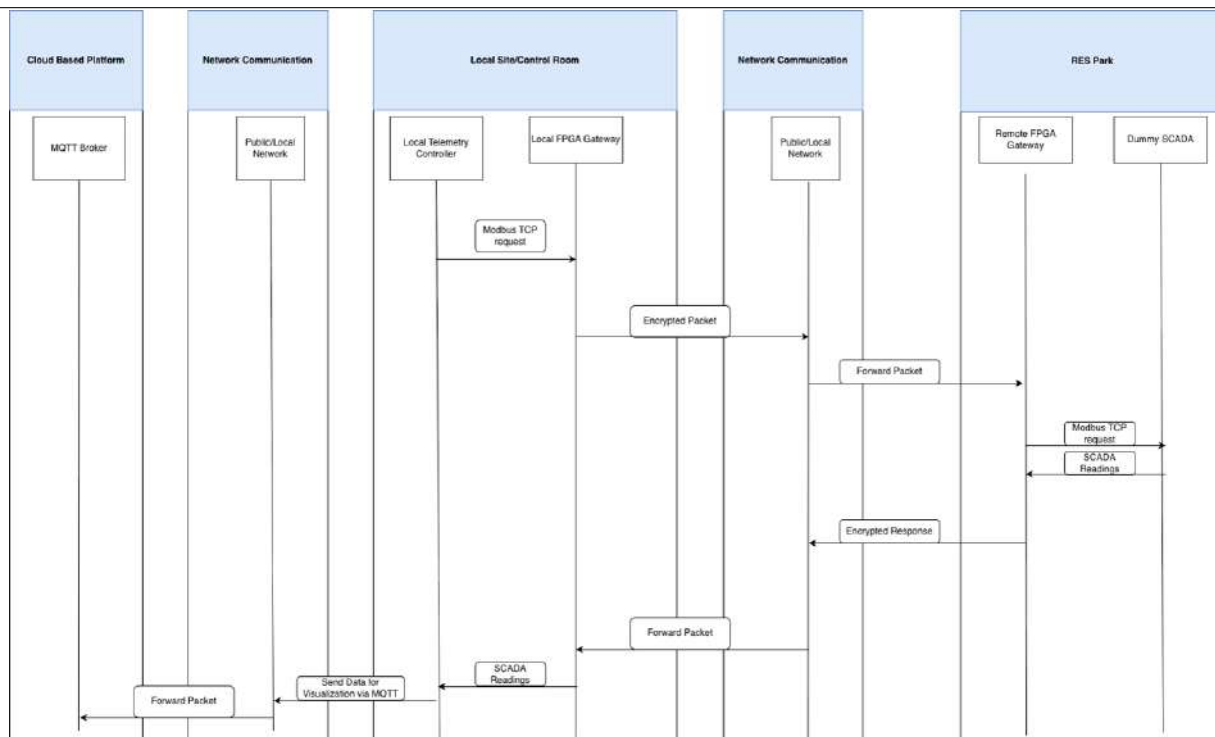


Figure 8.1: UC4 "As-Is" Smart Grid Communication Architecture

- **CERTH (Monitoring and Control Specialist):** Technical partner responsible for cross-infrastructure monitoring, security posture assessment, and validation of PQC integration across the UC4 deployment.
- **Smart Grid Control Centre (SCADA):** The central management system operated by MOH, responsible for aggregating telemetry per RES park, issuing control commands, and managing grid operations.
- **Gateways:** Intermediate network nodes that aggregate data from each RES SCADA and relay it to the monitoring platform. These are the primary integration points for PiQASO SDK deployment at MOH.
- **PiQASO SDK:** The software library providing ML-KEM and ML-DSA primitives for integration into gateway and endpoint software at MOH and MORE.
- **PQCaaS Backend:** The cloud-based PiQASO service that offloads cryptographic operations (key encapsulation, signing, ABE policy evaluation, homomorphic computations) for constrained devices.
- **Quantum-Resistant Trusted Computing Base (QR-TCB):** The PiQASO component that manages long-term cryptographic asset protection and enforces secure execution boundaries.

8.5 Challenges and Needs from PiQASO

The energy sector faces a unique combination of challenges in its transition to quantum-resistant cybersecurity. The following key challenges motivate UC4's design:

- Electrical smart grids span a wide network of legacy systems, constrained edge devices, and real-time operational requirements, making traditional system-wide cryptographic upgrades difficult and risky.
- Current RSA/ECC cryptography is applied at the gateway level, not at far-edge sensor level. Very resource-constrained devices that cannot run an OS are out of scope for PiQASO (the need is acknowledged but exceeds the project scope).
- Securing communications across diverse components without disrupting operational continuity requires stricter requirements for far-edge acceleration and the inclusion of side-channel countermeasures (CMs). Accelerated PQC versions with CMs will be tested in the second round of experiments.
- Authentication of a growing number of devices and users across untrusted or dynamic environments must be addressed while maintaining compliance with NIS2 and the Cyber Resilience Act.
- Cryptographic solutions must be resilient to side-channel attacks, support forward secrecy, and be adaptable to the fast-evolving PQC standards landscape. ABE is retained specifically for privacy-by-design analytics; other advanced primitives apply only where operationally justified.

PiQASO responds to these needs by offering a PQC-as-a-Service (PQaaS) model, enabling the sector to adopt post-quantum protection without requiring wholesale device replacement. This model allows computationally intensive cryptographic operations – such as key encapsulation, digital signing, and homomorphic encryption – to be securely offloaded to the cloud, while still ensuring end-to-end data protection. The energy sector also needs secure, policy-based access control mechanisms that can distinguish between user roles, locations, and device types, which PiQASO supports through attribute-based encryption (ABE). Additionally, the ability to perform encrypted computations enabled through homomorphic encryption allows energy providers to manage data such as usage statistics or fault predictions without exposing sensitive information.

8.6 “To-Be” Reference Scenario #1: Quantum-Secure Smart Grid Telemetry

Purpose within PiQASO: This reference scenario addresses MOH’s core operational need to transmit real-time smart grid telemetry – voltage, current, frequency, and active power – between the SCADA of a RES park and the gateway that aggregates this information. This needs to be enabled via quantum-resistant encrypted channels. The scenario demonstrates how PiQASO enables a quantum-secure communication backbone for critical grid operations, ensuring data integrity and confidentiality even against future quantum adversaries capable of breaking classical ECC and RSA cryptography. It validates the feasibility of replacing legacy classically-secured telemetry protocols with hybrid PQ-TLS connections, providing both a migration path and a future-proof architecture. **PiQASO Components Required for Instantiation:** The following PiQASO components are required to instantiate this reference scenario:

- **PiQASO Software Library (ML-KEM / ML-DSA):** Deployed on the gateway nodes to provide local PQC key encapsulation and digital signature capabilities. This enables the establishment of PQ-TLS mutual authentication sessions between the gateway and the SCADA without relying on the cloud backend for real-time signing.
- **Quantum-Resistant Trusted Computing Base (QR-TCB):** Deployed at the gateway layer to protect long-term PQC cryptographic material (private keys, certificates) and enforce secure execution of PQC operations, preventing key extraction even in physically exposed substation environments.

- **PQCaaS Backend:** Used for certificate management, algorithm negotiation, and offloading of computationally intensive cryptographic operations for legacy SCADA components that cannot host the full PiQASO SDK. Operates as a cloud-based PQC service endpoint accessed by both the SCADA and the gateway.
- **Crypto Assessment Toolkit (from WP2):** Monitors the cryptographic posture of the MOH infrastructure, tracks PQC migration progress, and provides tamper-evident audit logs for regulatory compliance with NIS2 and the Cyber Resilience Act.

System Architecture Integration: The PiQASO SDK is integrated into the software stack of the gateway, which acts as the primary encryption boundary the SCADA and the monitoring platform. The gateway is upgraded to initiate PQ-TLS sessions – using ML-KEM for key encapsulation and ML-DSA for mutual authentication – when communicating with the SCADA. The QR-TCB is embedded within the gateway hardware or trusted execution environment (TEE) to protect PQC key material at rest. The PQCaaS backend is deployed within the existing infrastructure and is accessed by the SCADA components that cannot host the full PiQASO SDK. The Crypto Assessment Toolkit is being used as an adhoc tool that can provide continuous visibility into the cryptographic health of the infrastructure.

8.7 “To-Be” Reference Scenario #2: Crypto-Agile Gateway

Purpose within PiQASO: This reference scenario addresses the long-term operational need of MOH to maintain cryptographic compliance without requiring hardware replacement every time a PQC standard evolves or is updated. SCADAs are long-lived physical assets, often operational for 15–25 years, making it impractical to replace hardware with each cryptographic transition. This scenario demonstrates how PiQASO’s crypto-agility capabilities allow a gateway to dynamically switch between supported PQC algorithm suites – for example, from ML-KEM-512 to ML-KEM-768, or co-establishing both ML-KEM and HQC keys to demonstrate algorithm-family diversity beyond lattice-based schemes – without service interruption. It validates the crypto-agility framework as a practical mechanism for ensuring long-term quantum readiness in critical infrastructure. PiQASO Components Required for Instantiation:

- **PiQASO Software Library (multi-algorithm):** The library must support multiple PQC algorithm suites simultaneously. The crypto-agility mechanism uses a runtime configuration layer that enables the gateway to select, negotiate, and activate a different algorithm suite based on policy signals from the central management system.
- **Crypto Assessment Toolkit / Trust Assessment Framework (TAF):** The TAF monitors the current cryptographic posture of the gateway node and triggers algorithm transitions when the existing suite falls below an acceptable security threshold or when a new NIST-approved standard is adopted. It provides the decision-making layer for automated crypto-agility.
- **QR-TCB:** Ensures that algorithm transitions are executed in a tamper-resistant manner, preventing adversarial manipulation of the algorithm selection process. The QR-TCB also manages key lifecycle events triggered by algorithm changes.
- **PQCaaS Backend:** Provides centralised key material generation and distribution for newly activated algorithm suites, enabling gateways to obtain fresh PQC credentials after a transition without the need for manual re-provisioning.

- **Attribute-Based Encryption (ABE) Module:** Enables policy-driven encryption of MORE's energy production data. Energy flow records and operational telemetry are encrypted under ABE policies that specify which MOH roles and system components may decrypt specific data categories, ensuring that commercially sensitive production data is only accessible to authorised parties.

System Architecture Integration: The crypto-agility capability is implemented as a software update to the gateway's PiQASO SDK integration layer. A new Algorithm Policy Manager module communicates with the TAF-based monitoring platform and the PQCaaS backend to receive algorithm update signals. The QR-TCB on each gateway node enforces the secure execution of algorithm transitions. The SCADA is updated to support multiple concurrent PQC algorithm suites in its TLS negotiation stack, enabling backward compatibility with gateways at different stages of transition. The full transition cycle – from policy signal to activated new suite – is targeted to complete in under two minutes without dropping active telemetry sessions.

8.8 “To-Be” Reference Scenario #3: Quantum-Secure Edge Device Onboarding

Purpose within PiQASO: The primary objective of this reference scenario is to establish a secure, zero-trust onboarding framework for newly deployed edge gateways within a Renewable Energy Source (RES) park infrastructure. As RES parks increasingly scale out distributed energy resources (DERs), edge gateways act as critical data aggregation and control hubs. Ensuring the integrity and authenticity of these devices at the point of initialization is paramount to protecting the broader grid from rogue node injections, man-in-the-middle (MitM) attacks, and unauthorized configuration alterations. This reference scenario directly addresses the lifecycle management phase of device commissioning, specifically defining how a pristine gateway can be securely bound to the PiQASO ecosystem in an untrusted field environment. To guarantee practical exploitability and validation, this scenario is scheduled for complete implementation during the second piloting phase of the PiQASO project.

The technical deployment will be dual-pronged, requiring deep architectural integration across two core endpoints:

- **The Edge Gateway Firmware:** Hosting the local cryptographic runtime environment responsible for provisioning verification, zero-touch key generation, and decentralized identity attestation.
- **The Companion Mobile Application:** Serving as the technician's localized, authenticated provisioning proxy. The app will orchestrate local ad-hoc communication channels (e.g., BLE, Wi-Fi Direct) to securely inject configuration profiles, verify hardware roots-of-trust, and bridge the gap between the isolated physical gateway and the central PiQASO Public Key Infrastructure (PKI).

PiQASO Components Required for Instantiation:

- **FN-DSA Signatures:** To future-proof the critical energy infrastructure against emerging quantum threats, the cryptographic foundation of this provisioning scenario abandons legacy asymmetric schemes (such as RSA or ECDSA) in favor of Post-Quantum Cryptography (PQC). Specifically, the scenario instantiates FN-DSA (Falcon, Fast-Fourier Lattice-based Compact Signatures over NTRU) as its primary digital signature algorithm to enforce post-quantum entity authentication and data integrity.

System Architecture Integration: The system architecture for this scenario is structured as a decentralized, three-tier post-quantum cryptographic framework consisting of the central PiQASO Public Key Infrastructure (PKI), the mobile provisioning application, and the embedded edge gateway layer. At the core, the PiQASO PKI operates a post-quantum Certificate Authority (CA) that issues and manages Falcon-signed root certificates and configuration profiles. The mobile application acts as an authenticated intermediary tier, establishing a secure local ad-hoc communication channel (such as Bluetooth Low Energy or Wi-Fi Direct) to discover the unprovisioned gateway in the field. The gateway itself forms the physical edge tier, utilizing an embedded cryptographic subsystem to run the Falcon verification engine and generate local post-quantum key pairs. During the initialization phase, the mobile app proxies the gateway's Falcon-signed Certificate Signing Request (CSR) to the central CA and subsequently injects the signed X.509 identity token back into the gateway's secure storage, establishing a quantum-resistant root of trust that enables secure upstream telemetry data flows over standard protocols like MQTT.

8.9 Reference Scenario User Stories and Metrics of Success

8.9.1 User Story #1 (Scenario #1): Secure Grid Telemetry Transmission

UC4.US1: Quantum-Resistant Telemetry Transmission in Smart Grid

As a Smart Grid Operator, I want to transmit real-time smart grid telemetry (voltage, current, frequency, and active power) between the RES park SCADA and the aggregating gateway using quantum-resistant encryption and mutual authentication so that data integrity and confidentiality are ensured even against future quantum adversaries capable of breaking classical ECC and RSA cryptography.

User Story Confirmation: The SCADA and the gateway establish a PQ-TLS session using the PiQASO Software Library for ML-KEM key encapsulation and ML-DSA for mutual certificate-based authentication. The gateway transmits grid telemetry continuously over the secured session. The gateway relies on its embedded Quantum-Resistant Trusted Computing Base (QR-TCB) to secure long-term PQC cryptographic material (private keys, certificates) and enforce secure execution of PQC operations. Concurrently, the Crypto Assessment Toolkit (from WP2) monitors the cryptographic health and posture of the infrastructure, providing tamper-evident audit logs for regulatory compliance with NIS2 and the Cyber Resilience Act.

User Story Workflow:

- 1. Key Protection & Handshake Initiation:** The gateway secures its local private keys within its hardware-embedded QR-TCB. Utilizing the local PiQASO Software Library, the gateway initiates a quantum-safe cryptographic handshake with the SCADA control center, advertising its capability to support hybrid PQC cipher suites (such as ML-KEM and ML-DSA).
- 2. Algorithm Selection & Certificate Presentation:** The SCADA negotiates the connection, selects the preferred quantum-resistant cipher suite, and presents its PQC certificate (signed with ML-DSA) issued by the PiQASO PKI. (Note: If the SCADA system contains legacy components that cannot host the full PiQASO SDK, it offloads this computationally intensive cryptographic negotiation to the cloud-based PQCaaS Backend).
- 3. Certificate Validation:** The gateway validates the server's PQC certificate against the trusted PiQASO root CA by verifying the ML-DSA signature. It interacts with the PQCaaS Backend to check the real-time certificate revocation status.

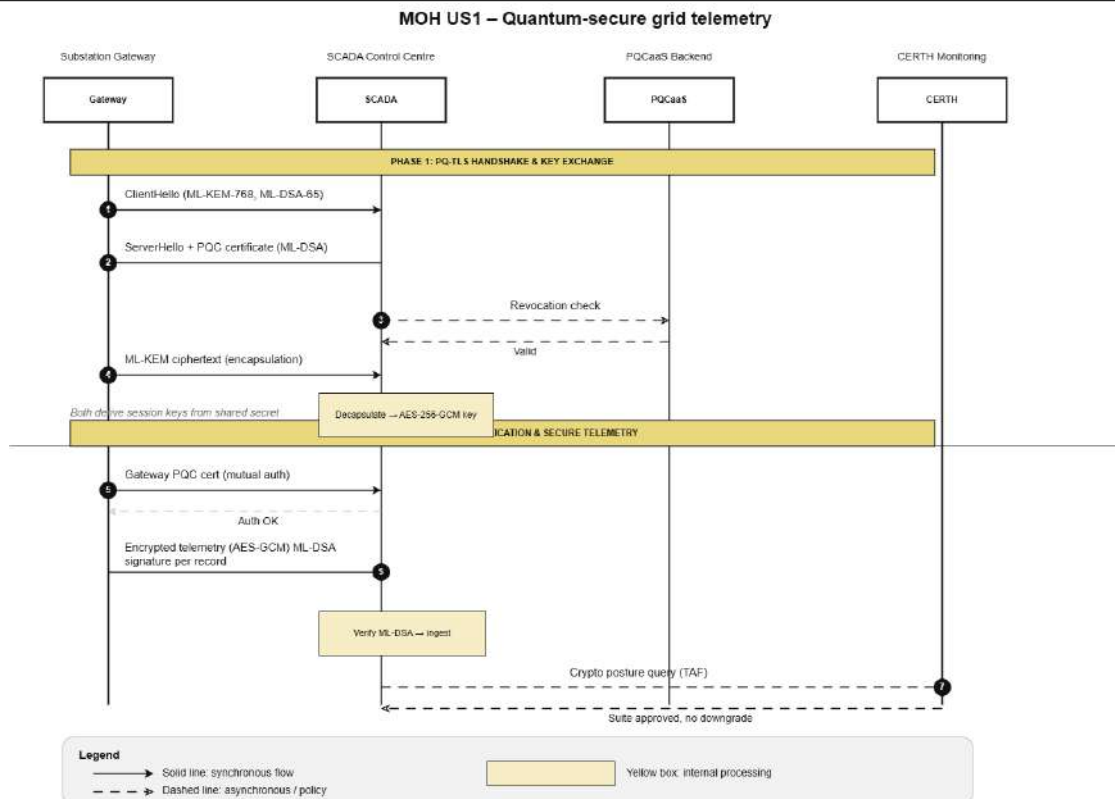


Figure 8.2: US1: Secure Grid Telemetry

4. **Key Encapsulation:** Utilizing the PiQASO Software Library, the gateway performs ML-KEM key encapsulation using the SCADA’s public key, producing a ciphertext encapsulation that is transmitted to the server.
5. **Symmetric Key Derivation:** The SCADA (or its integrated PQCaaS Backend proxy) decapsulates the ML-KEM ciphertext using its private key to obtain the shared session secret. Both parties derive symmetric AES-256-GCM session keys from this shared quantum-safe secret to establish the primary encryption boundary.
6. **Mutual Authentication Completion:** To complete the mutual authentication, the gateway presents its own ML-DSA PQC certificate to the server, which validates it back against the PiQASO PKI root.
7. **Secure Telemetry Transmission:** The gateway begins continuously transmitting the real-time grid telemetry (voltage, current, frequency, and active power) over the established AES-256-GCM encrypted channel, ensuring strict confidentiality across the communication backbone.
8. **Ingestion & Data Integrity Verification:** The gateway decrypts the payload, verifies the integrity of the received telemetry records, and safely ingests the data into the live grid monitoring dashboard.
9. **Cryptographic Auditing & Compliance Monitoring:** To verify that the active session maintains the approved PQC algorithm suite and that no cryptographic downgrade attacks have occurred, the Crypto Assessment Toolkit (from WP2) passively logs session metadata. The tool generates tamper-evident audit logs of the infrastructure’s cryptographic posture, allowing a verification service (such as CERTH) to periodically query the toolkit and audit NIS2/Cyber Resilience Act compliance without requiring gateways to generate heavy, non-trivial runtime signature attestations.

8.9.2 User Story #2 (Reference Scenario #2): Dynamic PQC Algorithm Switching (Crypto-Agility)

UC4.US2: Dynamic PQC Algorithm Update Without Service Interruption

As a *Utility Infrastructure Manager*, I want to *dynamically update the PQC algorithm suite used by a gateway without interrupting active telemetry sessions or requiring physical on-site intervention* so that *long-lived SCADA assets remain cryptographically compliant as PQC standards evolve or are updated over their 15–25 year operational lifespans*.

User Story Confirmation: The Crypto Assessment Toolkit / Trust Assessment Framework (TAF) detects that a gateway's current algorithm suite has been flagged for transition (e.g., due to a new NIST recommendation, policy update, or a need to co-establish non-lattice HQC keys to demonstrate algorithm-family diversity). An Algorithm Policy Manager module communicates with the TAF-based monitoring platform to execute a smooth, live algorithm transition. This transition is managed in a tamper-resistant manner by the QR-TCB and supported by the PQCaaS Backend to provision new key material. The full transition cycle from policy signal to newly activated suite completes in under two minutes without dropping active telemetry sessions or losing sensitive energy flow records.

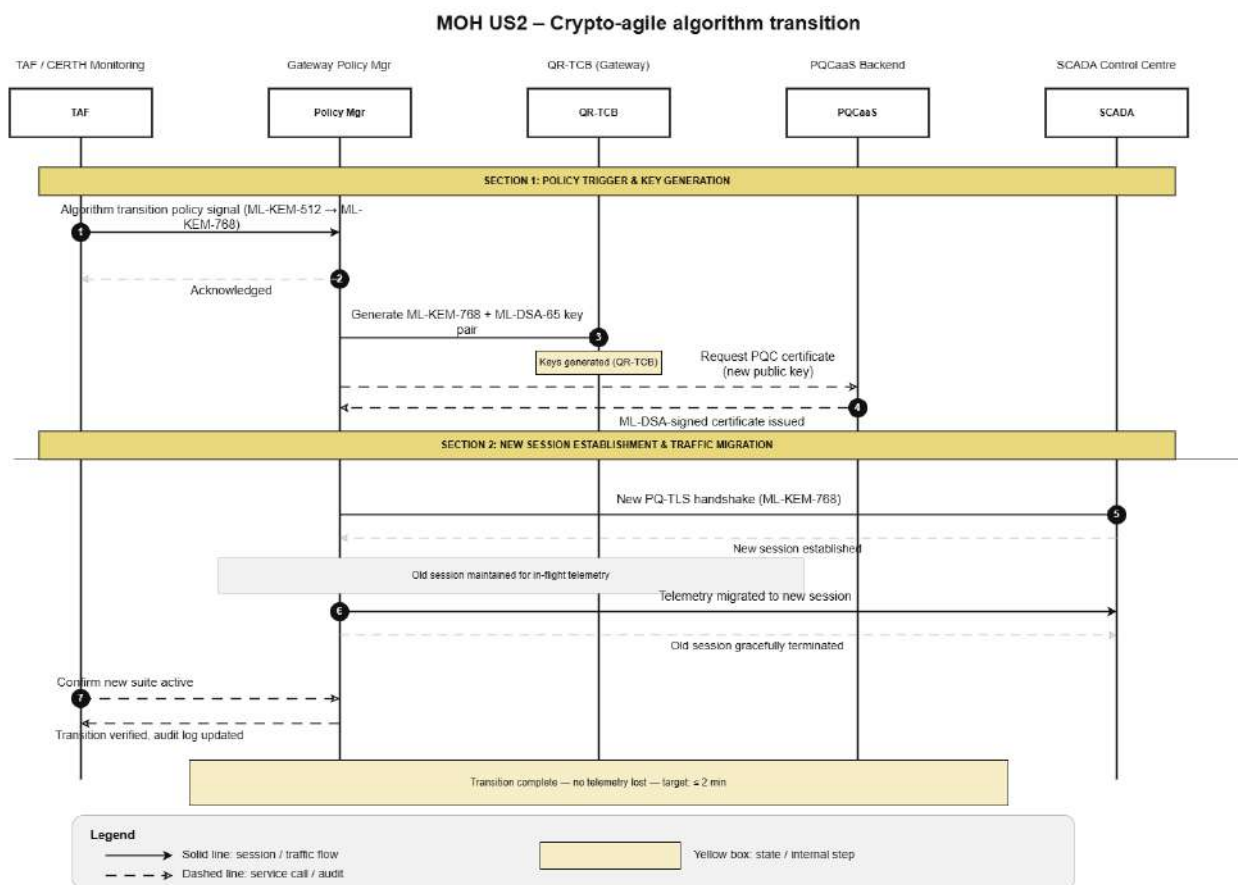


Figure 8.3: US2: Crypto-Agile Algorithm Transition

User Story Workflow:

- Posture Monitoring & Policy Trigger:** The Crypto Assessment Toolkit / Trust Assessment Framework (TAF) monitors the cryptographic posture of the gateway node. It identifies that the current suite falls below an acceptable security threshold or needs an update (e.g., transitioning from ML-KEM-512 to ML-KEM-768, or co-establishing HQC keys to achieve algorithm-family diversity be-

yond lattice-based schemes). The TAF issues an Algorithm Transition Policy signal to the gateway's Algorithm Policy Manager module.

2. **Secure Transition Initiation:** The Algorithm Policy Manager receives the signal and coordinates with the local QR-TCB to enforce a secure, tamper-resistant execution environment, preventing any adversarial manipulation of the algorithm selection process.
3. **Key Lifecycle Generation:** The QR-TCB triggers local key lifecycle events, automatically generating fresh public/private key pairs for the newly requested PQC suite (e.g., ML-KEM-768 or HQC) internally within its protected hardware boundary.
4. **Centralized Key and Credential Delivery:** The gateway interacts with the centralized PQCaaS Backend to obtain fresh PQC credentials. The PQCaaS backend processes the request and distributes a new PQC certificate bound to the newly generated public keys, signed by the PiQASO CA under the updated algorithm parameters without requiring manual device re-provisioning.
5. **Concurrent Session Negotiation:** Utilizing the multi-algorithm PiQASO Software Library, the gateway configuration layer initiates a new PQ-TLS session negotiation with the SCADA using the updated suite. The SCADA system utilizes its multi-algorithm TLS negotiation stack to maintain backward compatibility, keeping the existing legacy session alive for in-flight telemetry to guarantee zero service interruption.
6. **Graceful Session Migration:** Once the updated, quantum-safe session is fully established and verified, the gateway migrates active telemetry streams (voltage, current, frequency, and active power) to the new channel and gracefully terminates the old session.
7. **Policy-Driven Data Encryption (ABE):** The gateway routes the migrated data through its Attribute-Based Encryption (ABE) Module. This module enforces policy-driven encryption of energy production data, ensuring that energy flow records and operational telemetry are encrypted under ABE policies so that commercially sensitive production data is only accessible to authorized roles and verified system components.
8. **Auditing and Target Closure:** The TAF verifies the new session's active algorithm suite, confirms that the entire transition cycle completed successfully in under the target threshold of two minutes with zero data loss, and records the transition event into the tamper-evident audit log for NIS2 compliance reporting.

8.9.3 User Story #3 (Reference Scenario #3): Quantum-Secure Edge Device Onboarding

UC4.US3: Secure Onboarding of Edge Gateways via Mobile Companion App

As a Field Technician at a Renewable Energy Source (RES) Park, I want to onboard and securely provision newly deployed edge gateways into the quantum-secure communication framework using a companion mobile application proxy over an untrusted field environment so that these devices can be bound to the PiQASO ecosystem and act as authenticated data aggregation and control hubs, without risk of rogue node injections, man-in-the-middle (MitM) attacks, or unauthorized configuration alterations.

User Story Confirmation: A pristine gateway is initialized at a production site. The technician uses the Companion Mobile Application as a localized, authenticated provisioning proxy to establish an ad-hoc local communication channel (e.g., BLE, Wi-Fi Direct). The mobile app coordinates with the Gateway

Firmware and the PiQASO Public Key Infrastructure (PKI) to execute zero-touch key generation and verify hardware roots-of-trust. Upon verifying the gateway's Falcon-signed Certificate Signing Request (CSR), the post-quantum Certificate Authority (CA) issues a localized X.509 identity token. This token is injected back into the gateway's secure storage, establishing a quantum-resistant root of trust that enables secure upstream telemetry data flows.

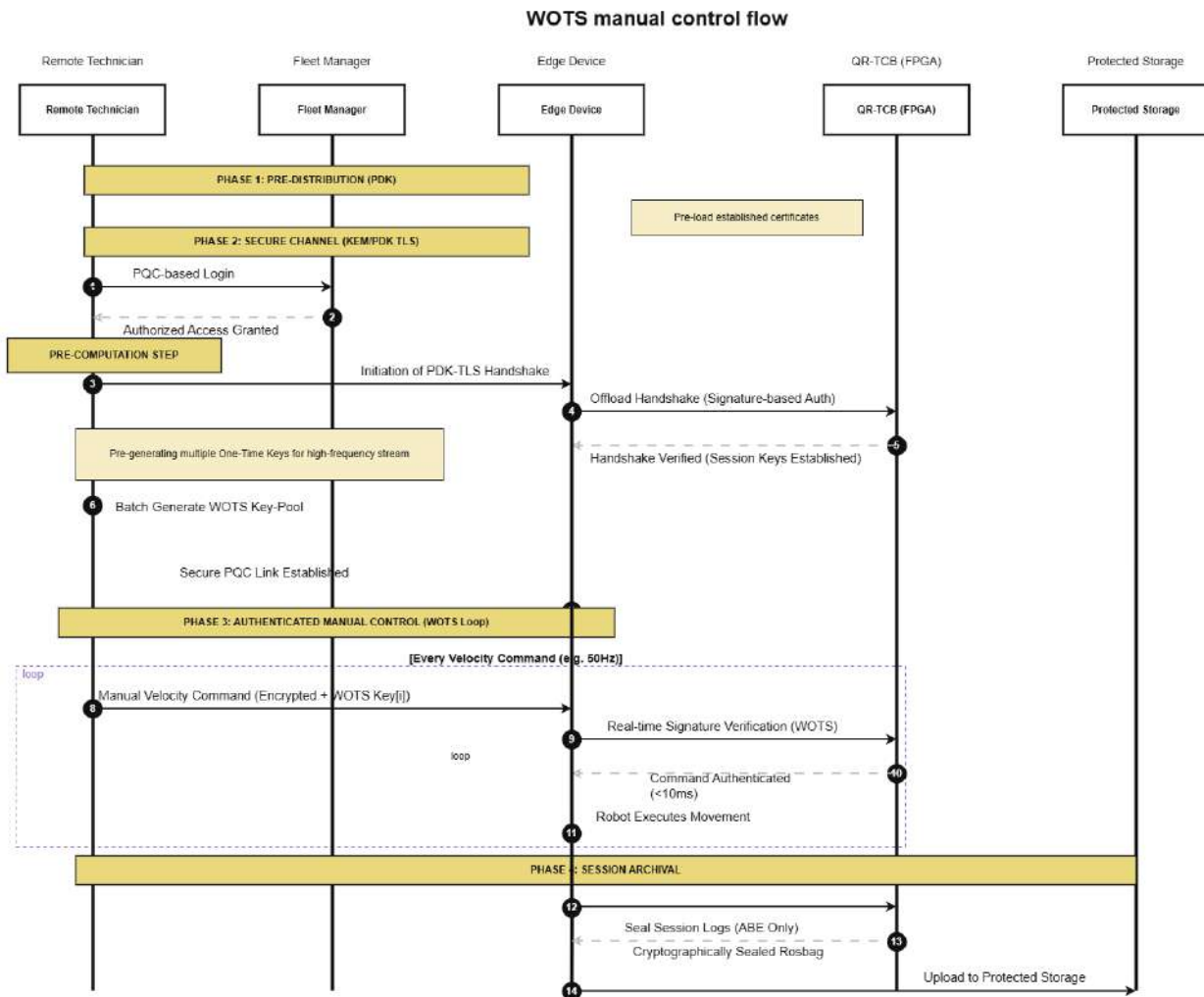


Figure 8.4: US3: Quantum-Secure Edge Device Onboarding

User Story Workflow:

- Ad-Hoc Discovery & Channel Establishment:** The field technician powers on the edge gateway at the RES park. The technician opens the Companion Mobile Application, which orchestrates a secure local ad-hoc communication channel (such as Bluetooth Low Energy or Wi-Fi Direct) to discover the unprovisioned gateway in the untrusted field environment.
- Configuration Profile Injection:** The mobile application pushes an initial, authenticated configuration profile issued by the self-hosted PiQASO PKI down to the gateway to initialize the onboarding session.
- Hardware Trust Enrolment & Key Generation:** The Gateway Firmware validates the configuration profile using its embedded cryptographic subsystem. Once validated, the gateway executes a zero-touch key generation sequence, establishing its local post-quantum key pair within its secure hardware storage layer.

4. **Falcon CSR Generation:** To ensure post-quantum entity authentication and data integrity while keeping payloads compact, the gateway utilizes Falcon (Fast-Fourier Lattice-based Compact Signatures over NTRU). It compiles a Certificate Signing Request (CSR) and signs it using its newly generated private Falcon key.
5. **Upstream Proxying via Mobile App:** Because the gateway is isolated during this initial commissioning phase, it transmits the Falcon-signed CSR over the local ad-hoc channel to the Companion Mobile Application. The mobile app acts as an authenticated intermediary tier, relaying the CSR upstream to the self-hosted PiQASO Public Key Infrastructure (PKI).
6. **CA Validation & Issuance:** The post-quantum Certificate Authority (CA) utilizes the gateway's public key to verify the Falcon signature on the incoming CSR. Upon validation, the CA issues a quantum-safe X.509 identity certificate, signs it using the root CA's Falcon key, and sends it back to the mobile application.
7. **Identity Injection & Boot Completion:** The mobile application pushes the Falcon-signed X.509 identity token down to the gateway. The Gateway Firmware injects the token into its secure local storage, establishing an immutable, decentralized identity attestation.
8. **Telemetry Streaming Activation:** With the post-quantum root of trust successfully established, the newly provisioned gateway activates its operational runtime environment. It begins securely aggregating SCADA data, streaming the encrypted upstream telemetry records to the monitoring platform over standard protocols.
9. **Auditing and Compliance Logging:** The central onboarding completion event is asynchronously logged within the Crypto Assessment Toolkit / Trust Assessment Framework (TAF) backend, providing an unforgeable, auditable record of the device's quantum-secure provisioning for NIS2 and Cyber Resilience Act compliance reporting.

8.9.4 Key Performance Indicators (KPIs)

Table 8.2 defines the KPIs used to measure the success of UC4, consolidated to consider both the Smart Grid Operator and Green Energy Producer operational contexts.

Table 8.2: Consolidated Key Performance Indicators (KPIs) for the Energy Use Case (UC4)

KPI	Description	Acceptance Criteria	Aligned Scenario Scope
Secure Channel Establishment Time	Operational round-trip time required to negotiate and establish a mutually authenticated, post-quantum session (PQ-TLS using ML-KEM and ML-DSA / Falcon) between a gateway and the SCADA control center.	$\leq 500\text{ms}$ per secure session establishment handshake.	Scenario #1 & Scenario #3: Enforces efficient handshake execution when building primary encryption boundaries.
Telemetry Throughput Retention	Percentage of nominal real-time smart grid telemetry throughput retained post-PQC integration, relative to the classical legacy baseline.	$\geq 95\%$ of telemetry data successfully transmitted within the designated time window (60 seconds).	Scenario #1 & Scenario #2: Guarantees that heavy PQC payloads do not impact continuous grid monitoring performance.

Crypto-Agility Transition Time	Maximum runtime duration needed to dynamically hot-swap between supported PQC algorithm families (e.g., lattice ML-KEM-512 to ML-KEM-768, or code-based HQC) on an active gateway node without service interruption.	≤ 2 min total execution time per live algorithm migration cycle.	Scenario #2: Measures the end-to-end performance of the local Algorithm Policy Manager and TAF trigger loop.
Authentication & Provisioning Failure Rate	Proportion of onboarding, initialization, or runtime authentication attempts that fail due to PQC-specific exceptions (e.g., lattice parameter rounding errors, key mismatches, or Falcon floating-point timing traps).	$\leq 0.1\%$ of all cryptographic validation and verification attempts.	Scenario #1, #2, & #3: Protects the grid infrastructure against Denial-of-Service (DoS) and provisioning timeout flags.
PQC Infrastructure Uptime	Global operational availability of the quantum-secured grid telemetry channels, the self-hosted PQCaaS Backend proxy, and the local QR-TCB execution runtime.	$\geq 99.5\%$ continuous availability during the scheduled pilot phase operations.	Scenario #1, #2, & #3: Ensures high availability across the physical edge, intermediary, and control center tiers.
Crypto Inventory & Posture Coverage	Completeness of the discovery, cataloging, and real-time posture monitoring of system primitives achieved by the Crypto Assessment Toolkit / Trust Assessment Framework (TAF).	$\geq 90\%$ of active infrastructure components continuously mapped and audited for NIS2 compliance.	Scenario #1 & Scenario #2: Guarantees passive monitoring visibility to support regulatory compliance without performance drag.

Chapter 9

PiQASO UC5.1 – Sector: Healthcare - Remote Patient Care

9.1 Use Case’s Description, Goals, and Objectives

Digital transformation in healthcare has led to the widespread adoption of connected medical devices, remote monitoring platforms, and cloud-integrated care infrastructures. These systems continuously process sensitive patient data, including physiological measurements, diagnostic information, and treatment records, making confidentiality, integrity, and authenticity critical requirements. Conventional cryptographic mechanisms currently deployed in healthcare infrastructures rely primarily on public-key schemes such as RSA and elliptic-curve cryptography, whose long-term security assumptions are threatened by advances in quantum computing [65, 46].

Quantum algorithms capable of efficiently solving integer factorization and discrete logarithm problems introduce a realistic future risk to widely used cryptographic primitives. Even before practical quantum computers reach maturity, the “harvest now, decrypt later” threat model highlights that encrypted medical data captured today may be decrypted retrospectively once quantum capabilities emerge [15]. Given the extended lifetime and regulatory sensitivity of healthcare data, this risk is particularly acute in the eHealth domain.

Healthcare IoT environments amplify these concerns. Edge devices often operate under constrained resources, are deployed in physically accessible locations, and communicate over heterogeneous networks. Secure device onboarding, authenticated firmware updates, and protected telemetry transmission are therefore essential components of a trustworthy infrastructure. Current approaches may struggle to provide long-term resilience, especially when devices are expected to remain operational for many years without hardware replacement [61]. Transitioning toward quantum-resistant mechanisms is thus not only a forward-looking security measure but also a practical requirement for lifecycle risk management.

Recent standardization efforts toward post-quantum cryptography aim to address these vulnerabilities by introducing algorithms designed to withstand both classical and quantum adversaries [49]. Integrating such mechanisms into healthcare systems enables the protection of sensitive data assets against emerging computational threats while maintaining trust in digital service delivery. However, the adoption of these techniques in real deployments requires careful evaluation of performance, interoperability, and operational feasibility - particularly within resource-constrained edge environments.

Within this context, the PiQASO initiative explores how post-quantum cryptographic techniques can be applied to real-world eHealth scenarios. By investigating secure onboarding, protected communication over MQTT-based infrastructures, and trusted update distribution using FPGA-accelerated cryptographic capabilities, the project contributes to understanding how quantum-resilient security can be integrated

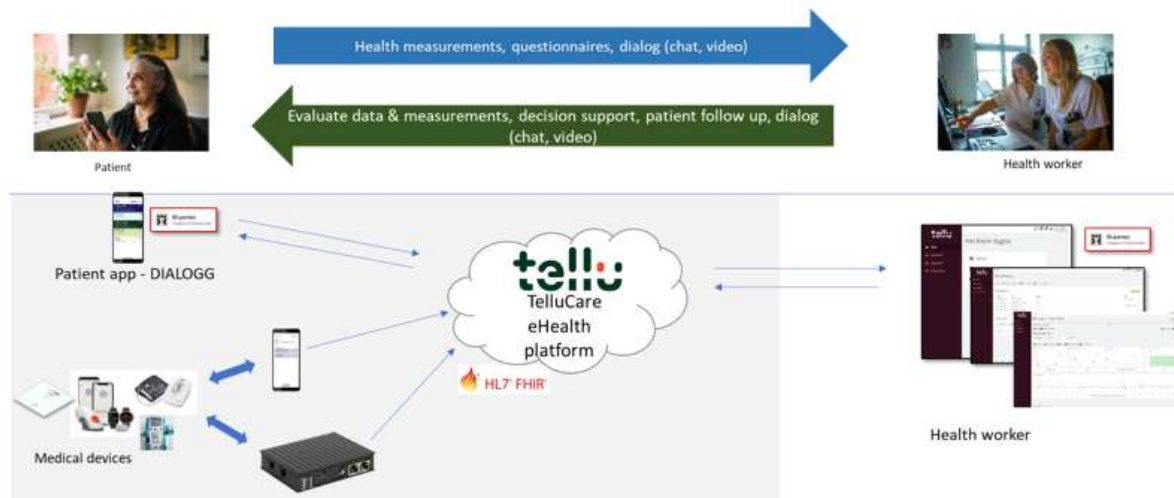


Figure 9.1: TelluCare Remote Patient Monitoring architecture and workflow

into platforms such as those used in the Tellu ecosystem. This work provides both practical deployment insights and measurable evaluation outcomes, supporting the broader transition toward future-proof healthcare security architectures.

This pilot is focusing on the TelluCare Remote Patient Monitoring system, which facilitates remote assistance to follow up on patients suffering from chronic diseases such as diabetes and COPD, or other diseases where patients can at least partly stay in their own home, such as COVID-19. In general, the system enables patients to stay home during treatment and care, and to foster self-care. This allows health services to utilize the trained medical personnel more efficiently.

The pilot revolves around the Tellu Personal Health Gateway (PHG), which is an IoT gateway device deployed in a patient's home. The PHG is responsible for collecting data from various medical sensors and sending them to the cloud platform TelluCare for further processing. The TelluCare service allows health personnel to access the data to perform remote patient monitoring, and to record the data and events in the patient's electronic health journal. The platform also normalizes the data according to standard eHealth ontologies to allow analysing the data in various ways and to easily integrate with other eHealth actors.

As the figure 9.1 indicates, the pilot executes across the IoT edge and cloud space. Medical devices and sensors on the IoT level is connected to the PHG, which forwards events and measurements from the devices to the TelluCare platform. The health personnel can then access the provided data and follow up the patient based on provided measures.

The PHG itself has two isolated environments, the management environment and the application environment. In the administration environment, the PHGs underlying software and configuration are updated and managed. The application environment is responsible for the handling of the patient data, as well as managing the application-specific components that access this data. This is illustrated in the figure 9.2.

The Tellu PHG, as shown in 9.3, applies an industrial gateway built on IoT-GATE-iMX8 - Industrial IoT Gateway from Compulab. This is a powerful ARM-based gateway that supports several possibilities for connections and data communication. In particular, the gateway has a modular architecture that allows it to be configured to support many different types of standardized and proprietary connections. It can, for example, integrate medical technical equipment with both standard protocols like BLE and with more proprietary protocols and equipment that can only be connected via USB port, serial port, Ethernet, Wi-Fi, etc

The physical gateway comes as standard with 3 USB, 2 Ethernet ports and a serial port (which can be set

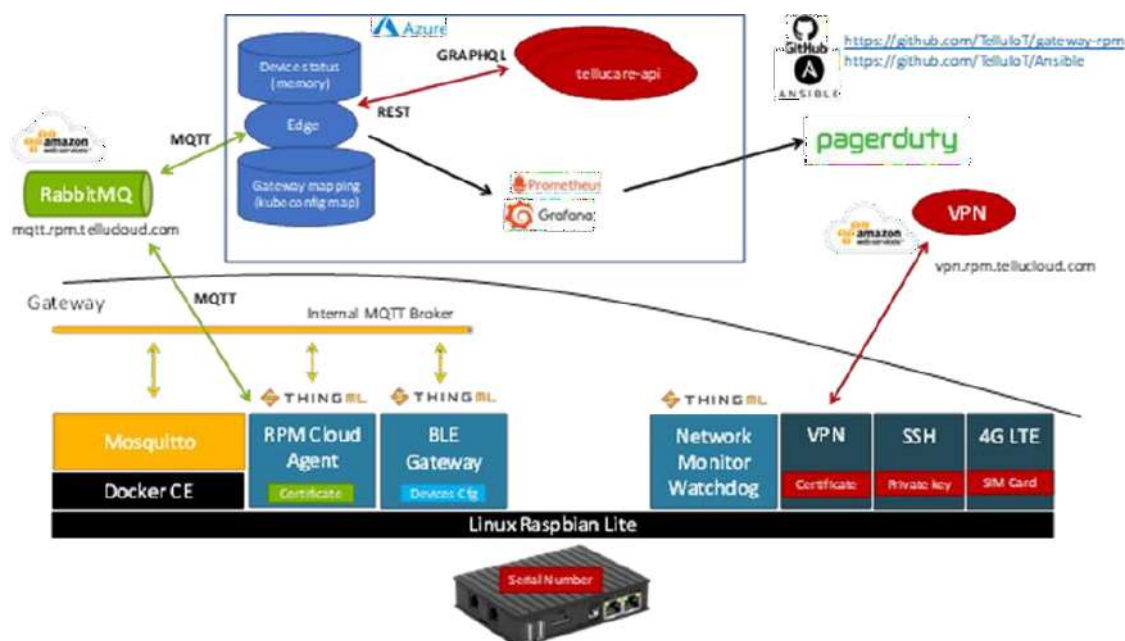


Figure 9.2: The Management (right)- and Administration (left) environments of the PHG



Figure 9.3: Tellu PHG - IOT-GATE-iMX8

up according to either RS232 or RS485). You can connect both USB and Ethernet hubs so that you can have many PHGs connected via USB and or Ethernet. Furthermore, the physical unit can be expanded to set up several simultaneous serial ports (RS485 and or RS232) for connecting the medical device. As mentioned, the physical unit is modular and has a set of available modules that provide a further wide range of I/O support. This is summarized in the list below:

- PCI Express (Primary mini-PCIe socket)
- USB (USB2.0 ports, type-A connectors)
- I/O for Debugging (serial console via UART-to-USB bridge, micro-USB connector)
- Serial (RS485 (2-wire) / RS232 port, terminal-block)
- Analog 4–20mA input
- Digital I/O: 4x digital outputs + 4x digital inputs 24V, støtter EN 61131-2, (isolated, terminal block connector)
- PoE, extra 100Mbps Ethernet port with PoE capacity (powered device)
- Expansion connector for add-on boards: 2x SPI, 2x UART, I2C, and 12x GPIO (General purpose I/O, that can be applied for more proprietary connections)

The most common medical device models we support are described in the table 9.1.

Table 9.1: Medical device models

Device Type	Model
Pulse oximeter	Nonin Pulse Oximeter 3230
Blood pressure monitor	A&D Blood pressure monitor UA651BLE
Scale	A&D Medical precision health scale UC352BLE
Thermometer	A&D Medical precision digital thermometer UT201BLE
Blood glucose meter	Contour Next One NEXT
Fitness wearable	Fitbit

Remote patient monitoring solutions set strict requirements with respect to security, privacy, and compliance. These requirements need to be ensured throughout the end-to-end value chain, which includes a set of components and data processing that are distributed across the IoT, edge, and cloud space. This introduces significant complexities, and the need for Tellu to enable significant scaling in terms of large-scale deployments of RPM services distributed in potentially hundreds of thousands of patient home increases these complexities substantially. The main challenges and needs that will be addressed for the Tellu use case are:

- Efficient and trustworthy onboarding: As an eHealth service provider, onboarding, including reliable authentication, is the first step. It is important to restrict access to only authorised and reliable actors. For the RPM use case, a compromised software version can turn a trusted device into a liability; it is therefore desirable to be able to verify the integrity of a device at runtime. With a growing customer base, a simple and scalable onboarding process for devices is vital to ensure the timely fulfillment of customer orders at acceptable costs. Through the PiQASO project, this demonstrator is exploring ways to provide secure and efficient onboarding of patients as well as IoT and edge components at runtime.

- **Trustworthy transmission of data:** After the onboarding process is completed, the patient can begin performing health measurements at home using the supported medical devices, as described in table 9.1. The collected measurements are first transmitted from the medical device to the Tellu Personal Health Gateway (PHG), typically over Bluetooth Low Energy (BLE). The PHG then forwards the data to the TelluCare cloud platform using the MQTT protocol, where the information is processed, stored, and made available for further analysis. Authorised healthcare personnel can securely log into the TelluCare services to access the patient's medical data, enabling continuous remote monitoring and timely clinical follow-up. Since this data is highly sensitive and directly related to patient safety, it is crucial to ensure that it cannot be intercepted, altered, or compromised during transmission between the devices, the gateway, and the cloud platform. Therefore, trustworthy and secure data transfer mechanisms are essential to guarantee the confidentiality, integrity, and safe storage of medical information throughout the entire end-to-end communication chain.
- **Robust and trustworthy scaling:** Large-scale deployments are complicated to manage. In Remote Patient Monitoring services, gateways and medical devices may be stowed away for extensive periods of time. When they come online again it is important that they are updated in a timely manner, especially if critical updates have been released while the device was disconnected. As a service provider, it is also important to have a good overview of metrics and usage of devices deployed to customers for robust and efficient operation. Through PiQASO we will explore robust and secure management of a large fleet of PHGs and connected devices in ubiquitous and volatile environments, utilizing the PiQASO components.

9.2 PiQASO Functionalities

In alignment with the objectives of the PiQASO project, this use case investigates the integration of selected post-quantum cryptographic (PQC) functionalities within the TelluCare Remote Patient Monitoring ecosystem. The goal is to evaluate how these mechanisms can enhance security across key operational processes, including device onboarding, medical data transmission, and software update management in an IoT–edge–cloud healthcare setting. The following table 9.2 provides an overview of the selected functionalities, their role within the pilot deployment, and the planned reporting periods for their implementation and evaluation.

Table 9.2: PiQASO Functionalities to be evaluated for eHealth usecase

PiQASO Functionality	Description and Purpose in the Use Case	RP1	RP2
PQ Signatures	Signature used in onboarding secret generation.	✓	
PQ-TLS	Mutual authentication gateway - Backend.	✓	
QR-TCB & HW Acceleration	Utilization of FPGA-based hardware acceleration to offload PQC primitives, ensuring real-time performance.	✓	✓
Over-The-Air Updates	OTA utilizing Dilithium/FALCON signatures (update server signs firmware; gateway verifies signature).		✓
Functional Encryption	i) FE on numerical patient data ii) utilizing CAT flavor of FE weakening the trust assumptions on the Tellu server (nice to have).		✓
ABE	For securing data stored in the cloud DB after parsing.		✓

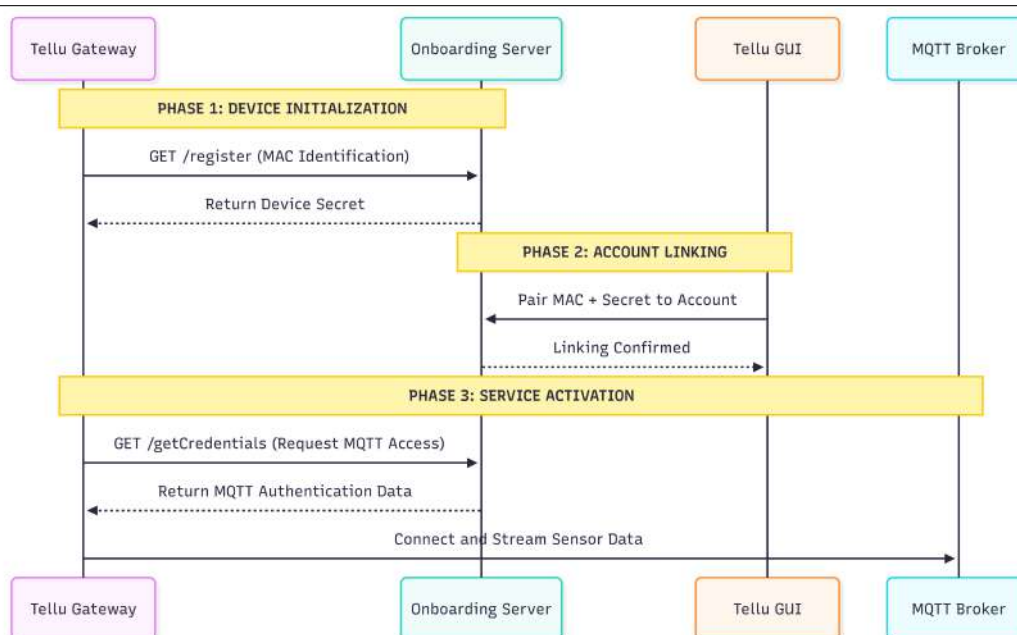


Figure 9.4: AS_IS onboarding of PHG to Tellu cloud sequence diagram

9.3 "As-is" Scenario

In this section we will describe the flow along with the process for all the three use cases that we have described above.

9.3.1 Efficient and trustworthy onboarding

In the current TelluCare Remote Patient Monitoring system, onboarding of the Personal Health Gateway (PHG) is a necessary prerequisite before the gateway can securely transmit medical measurements to the cloud platform. The onboarding process ensures that only authorised gateway devices can connect to the TelluCare infrastructure and obtain the credentials required for operational communication. The existing onboarding workflow comprises three main phases: device initialisation, account linking, and service activation, as shown in 9.4.

Phase 1 – Device Initialisation

Step 1: When the PHG is powered on for the first time, it automatically sends a GET /register request to the Onboarding Server, providing its MAC address as the device identifier. This step is triggered at boot time without requiring any manual interaction from the customer.

Step 2: The Onboarding Server generates a unique device secret associated with the gateway, stores the MAC–secret pair in its database, and returns the secret to the gateway. The gateway stores this secret locally. At this point, the device is registered as a legitimate gateway but is not yet associated with any customer account or TelluCare domain.

Phase 2 – Account Linking

Step 3: The healthcare provider initiates domain onboarding by submitting the gateway's MAC address and the previously issued secret through the TelluCare graphical user interface. This manual step associates the physical gateway deployed in the patient's home with the correct customer account in the platform.

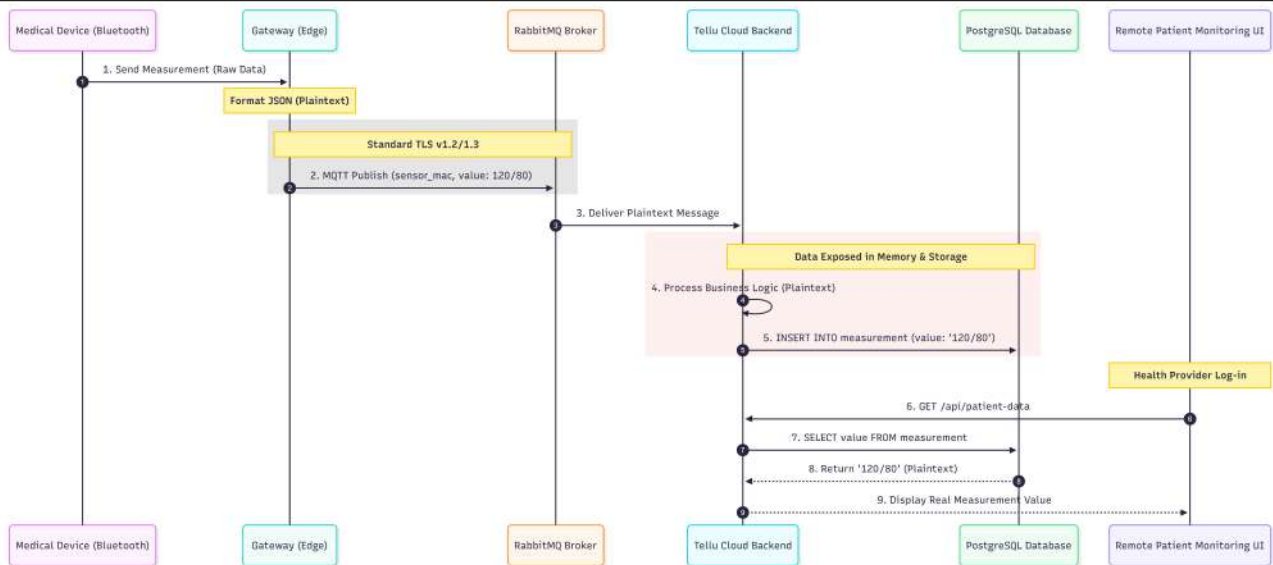


Figure 9.5: AS_IS sequence diagram for data transfer from PHG to Tellu services

Step 4: The Onboarding Server confirms the account linking and notifies the Tellu GUI that the association has been completed successfully.

Phase 3 – Service Activation

Step 5: On its next boot following successful account linking, the gateway sends a GET /getCredentials request to the Onboarding Server, using its stored MAC address and secret to authenticate itself and request MQTT access credentials.

Step 6: The Onboarding Server verifies that the provided MAC address and secret match the stored registration record and that the gateway status is valid. Upon successful verification, the server generates and returns the required MQTT authentication data to the gateway.

Step 7: The gateway stores the received MQTT credentials locally and uses them to connect to the MQTT Broker, completing the onboarding workflow. The PHG is now fully onboarded into the customer domain and ready to stream patient measurement data to the TelluCare cloud infrastructure.

This current onboarding approach provides a basic mechanism for device registration and access control. However, the reliance on plain HTTP for registration requests and the absence of cryptographic authentication of the gateway's identity represent limitations that become increasingly significant as the system scales to large deployments across many patient homes. These limitations motivate the transition to a post-quantum secure onboarding architecture described in the To-Be scenario.

9.3.2 Trustworthy transmission of data

After the onboarding process has been successfully completed, the Personal Health Gateway (PHG) operates as the central communication hub in the patient's home environment. In the current TelluCare Remote Patient Monitoring system, the PHG collects medical measurements from a set of supported devices, such as blood pressure monitors, pulse oximeters, thermometers, weight scales, and glucose meters. These devices enable patients suffering from chronic diseases to perform regular health measurements from home, supporting continuous clinical follow-up without requiring frequent physical visits to healthcare facilities. The data transmission process follows a sequential flow from the medical device at the edge, through the gateway, and into the TelluCare cloud backend, as illustrated in 9.5.

Step 1 – Measurement Acquisition: The patient performs a health measurement using a supported medical device. The device transmits the recorded raw data to the Gateway over Bluetooth, where it is formatted into a JSON payload for onward transmission.

Step 2 – MQTT Publish: The Gateway publishes the formatted measurement as an MQTT message to the RabbitMQ Broker over a Standard TLS v1.2/1.3 secured connection. The published payload contains the sensor MAC address and the recorded measurement value (e.g. `sensormac, value: 120/80`). It is important to note that at this stage, the transmitted data carries no direct patient identifiers. The sensor MAC address serves only as a device-level reference and cannot be associated with any individual patient during transit.

Step 3 – Message Delivery: The RabbitMQ Broker delivers the message to the Tellu Cloud Backend for processing.

Step 4 – Business Logic Processing: The Tellu Cloud Backend processes the received measurement by executing the relevant business logic, such as range validation and anomaly detection. It is at this stage that the patient association is performed: the backend maps the incoming device identifier to the corresponding patient record, linking the measurement to the correct individual within the TelluCare platform.

Step 5 – Database Storage: The processed and now patient-associated measurement is inserted into the PostgreSQL Database (e.g. `INSERT INTO measurement (value: '120/80')`), where it is stored for subsequent retrieval and clinical review.

Step 6 – Healthcare Provider Access: An authorised healthcare provider logs into the Remote Patient Monitoring UI to access patient data.

Step 7 – Data Query: The UI sends a `GET /api/patient-data` request to the Tellu Cloud Backend, which in turn queries the PostgreSQL Database with a `SELECT value FROM measurement` statement.

Step 8 – Data Retrieval: The PostgreSQL Database returns the stored measurement value (e.g. `'120/80'`) to the Tellu Cloud Backend.

Step 9 – Display: The Tellu Cloud Backend forwards the measurement value to the Remote Patient Monitoring UI, where it is displayed to the healthcare provider for clinical review and decision-making.

The current data transmission approach is functional and provides a reasonable level of protection for the scale at which TelluCare currently operates. The anonymisation of data during transit means that intercepted messages cannot be directly linked to individual patients, which reduces the immediate privacy risk during communication. However, as TelluCare scales to larger deployments and the threat landscape continues to evolve, introducing encryption of measurement data during both transmission and storage becomes increasingly important. This is particularly relevant in the context of future quantum-capable adversaries, which could undermine the classical cryptographic protections currently in place. These considerations motivate the To-Be scenario, which introduces end-to-end encryption of medical measurements from the point of acquisition at the gateway through to secure storage in the cloud backend.

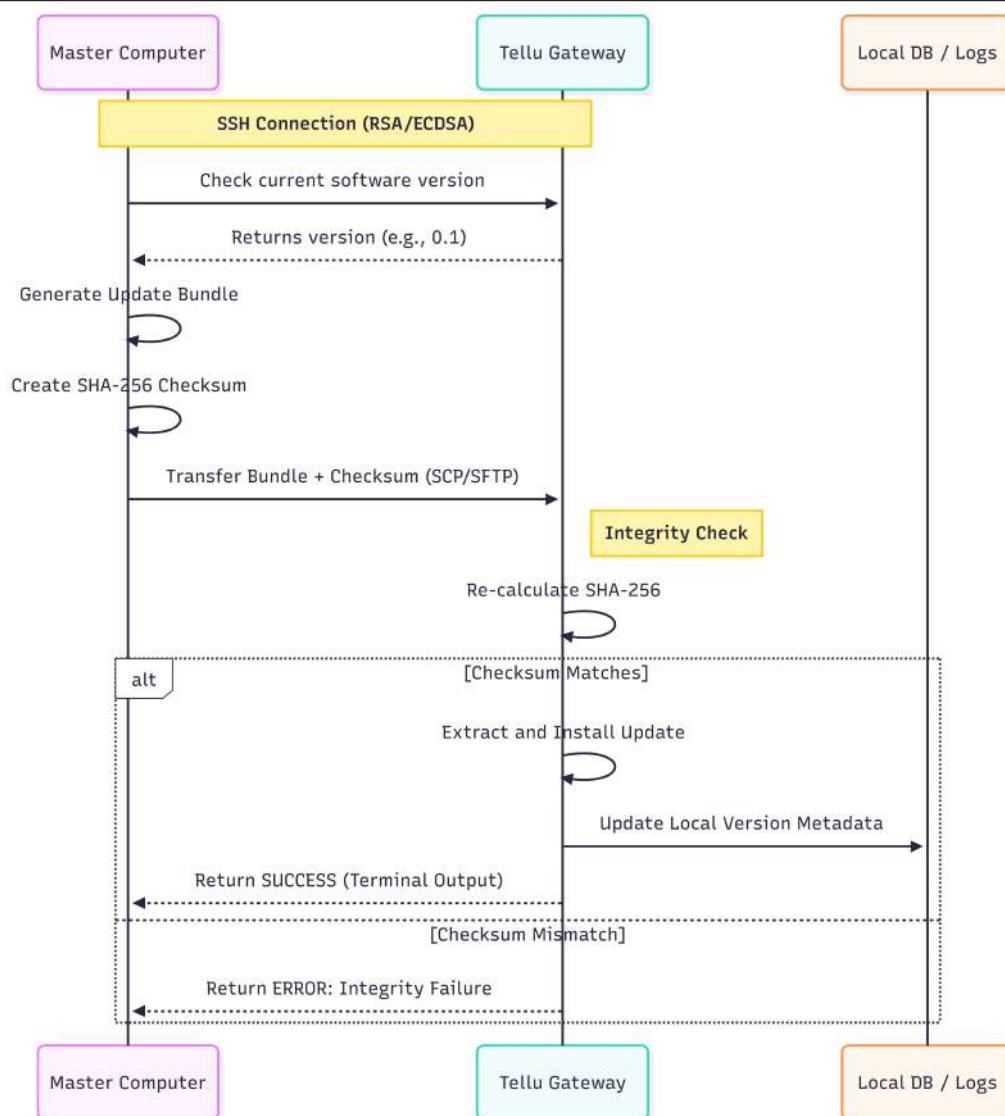


Figure 9.6: AS-IS process for the update of the gateway sequence diagram

9.3.3 Robust and trustworthy scaling

In the current TelluCare Remote Patient Monitoring deployment, software maintenance and updates of the Personal Health Gateways (PHGs) are performed through remote update procedures managed by Tellu. Since the gateways are distributed across many patient homes, it is essential to ensure that deployed devices can be kept up to date with the latest software versions, including functional improvements and security patches. The existing update approach is based on remote access via Secure Shell (SSH) connections, with Ansible scripts used to automate and orchestrate the deployment process across the gateway fleet. The full update workflow is illustrated in 9.6.

Step 1 – Version Check: The Master Computer establishes an SSH connection to the Tellu Gateway using RSA/ECDSA authentication and sends a request to check the current software version running on the gateway. The gateway returns its current version (e.g. 0.1), allowing the Master Computer to determine whether an update is required.

Step 2 – Update Bundle Generation: If an update is needed, the Master Computer generates the software update bundle locally, packaging the necessary changes and new software components.

Step 3 – Checksum Creation: The Master Computer computes a SHA-256 checksum of the generated update bundle. This checksum will be used by the gateway to verify the integrity of the received package after transfer.

Step 4 – Bundle Transfer: The Master Computer transfers the update bundle together with its SHA-256 checksum to the Tellu Gateway over a secure SCP/SFTP channel established within the SSH connection.

Step 5 – Integrity Check: Upon receiving the update bundle, the gateway independently re-calculates the SHA-256 checksum of the received package and compares it against the checksum provided by the Master Computer.

Step 6a – Successful Update [Checksum Matches]: If the checksums match, the gateway extracts and installs the update package. Following successful installation, the gateway updates its local version metadata in the Local DB / Logs and returns a SUCCESS confirmation to the Master Computer via terminal output.

Step 6b – Failed Update [Checksum Mismatch]: If the checksums do not match, the gateway rejects the update package and returns an ERROR: Integrity Failure message to the Master Computer, preventing the installation of a corrupted or incomplete update.

The current update mechanism provides a functional and structured approach to remote software maintenance, with basic integrity verification through SHA-256 checksums and reliable automation through Ansible. However, the use of classical RSA/ECDSA authentication for SSH connections and the absence of cryptographic signing of update packages mean that the authenticity of the update itself cannot be formally verified at the gateway level. As the TelluCare fleet scales and the threat landscape evolves, particularly with the prospect of quantum-capable adversaries, strengthening the integrity and authenticity guarantees of the update process becomes increasingly important. These considerations motivate the To-Be scenario, which introduces post-quantum digital signatures and blockchain-based state anchoring to provide a more robust and trustworthy software update mechanism.

9.4 Components and Roles Considered

The TelluCare Remote Patient Monitoring use case involves a set of distributed components operating across the patient home environment, the edge gateway layer, and the cloud platform. To support secure onboarding, reliable medical data transmission, and robust software maintenance, several key system components, communication mechanisms, and operational roles are considered within the scope of this pilot. The most relevant components and roles are summarised below.

- **Tellu Personal Health Gateway (PHG)** The main edge component is deployed in the patient's home, responsible for collecting measurements from connected medical devices and forwarding them to the TelluCare cloud platform.
- **Medical Devices and Sensors** Patient-operated devices such as blood pressure monitors, pulse oximeters, thermometers, glucose meters, and scales, which transmit medical measurements to the PHG, typically via Bluetooth Low Energy (BLE).

- **TelluCare Cloud Platform** Cloud backend receives measurement data from gateways, processes and stores the data, and makes it available for remote patient monitoring by authorised healthcare services.
- **Onboarding Server and Database (Project Setup)** Dedicated onboarding infrastructure used in this project to register gateways at first boot, generate device secrets, and store MAC–secret pairs for later authentication and credential provisioning.
- **RabbitMQ / MQTT Communication Infrastructure** Messaging system enabling data transmission between the PHG and the TelluCare platform using the MQTT protocol, including the issuance of MQTT credentials during domain onboarding.
- **Service Provider / System Operator** Responsible for managing the deployment and operation of gateways, including onboarding support, communication setup, and software maintenance.
- **Remote Software Update Mechanism (SSH + Ansible)** Current update approach, in which online gateways are maintained via automated Ansible scripts executed over SSH connections.
- **Key Data Exchanges** Gateway registration requests, secret provisioning, MQTT credential issuance, transmission of medical measurements from devices through the PHG to the cloud, and deployment of remote software updates.

9.5 Challenges and Needs from PiQASO

Remote Patient Monitoring systems operate across patient homes, edge gateways, and cloud-based healthcare services. While the current solution enables efficient follow-up of patients outside clinical settings, it also introduces significant challenges related to trust, privacy, and long-term security. Medical and patient data is highly confidential, and any compromise may have severe consequences for patient safety, trust in healthcare services, and regulatory compliance. In addition, evolving cyber threats and the emergence of quantum computing increase the need for stronger and future-proof security mechanisms across the entire IoT–edge–cloud chain.

- **Onboarding and Device Authentication**
 - A key challenge is ensuring that only legitimate and uncompromised gateway devices are able to join the TelluCare infrastructure.
 - The current onboarding approach relies on device secrets and credential provisioning over plain HTTP connections, which may be vulnerable if credentials are intercepted, leaked, or devices are impersonated.
 - There is a need for stronger authentication mechanisms and secure onboarding procedures that can scale to large deployments while maintaining trust in the identity and integrity of every enrolled device.
 - The long-term security of traditional cryptographic algorithms used for authentication is directly threatened by the advent of quantum computing, motivating the need for post-quantum resilient authentication and key exchange mechanisms that remain secure against future quantum-capable adversaries.
- **Trustworthy Transmission of Medical Data**

- Medical measurements transmitted from patient homes are highly sensitive and must remain confidential and protected against manipulation throughout the full communication chain.
- In the current system, measurement data is anonymised during transit in the sense that transmitted payloads carry no direct patient identifiers, which provides a reasonable level of privacy protection at the current scale of deployment. However, as the TelluCare deployment grows and the threat landscape evolves, this approach alone is insufficient to guarantee the confidentiality of sensitive health data.
- Even when communication channels are secured, data is typically decrypted for processing and storage in backend systems, increasing the risk of exposure through cloud-side attacks or insider threats. There is therefore a need for end-to-end confidentiality guarantees that keep medical measurements encrypted throughout transmission and storage, reducing unnecessary exposure of plaintext data while still enabling clinical monitoring and analysis.
- Classical encryption schemes currently used to protect data in transit and at rest may become vulnerable to quantum attacks in the foreseeable future, further motivating the need for post-quantum secure encryption mechanisms that provide long-term protection of sensitive patient information.

• **Robust and Trustworthy Software Updates**

- Maintaining a large fleet of gateways over time is challenging, especially since devices may remain offline for extended periods and miss critical security patches, resulting in inconsistent patch levels across the deployed fleet.
- The software update process itself represents a significant attack surface, as adversaries could attempt to inject malicious or unauthorised software onto gateways during distribution or deployment.
- While the current update mechanism provides basic integrity verification through SHA-256 checksums, it does not provide cryptographic authentication of the update source. This means that a correctly checksummed but malicious update package cannot be distinguished from a legitimate one at the gateway level, leaving the update process vulnerable to supply chain attacks.
- There is a need for mechanisms that provide strong cryptographic verification of both the integrity and the authenticity of update packages before installation, ensuring that gateways can independently confirm that received updates originate from an authorised source and have not been tampered with.
- As with other components in the system, the classical signing schemes currently used for SSH authentication are threatened by quantum computing advances, motivating the adoption of post-quantum digital signature schemes for update package signing and verification that remain secure over the long operational lifetime of deployed gateways.

9.6 "To-Be Scenarios

The PiQASO-enabled To-Be scenarios introduce enhanced security mechanisms across the TelluCare Remote Patient Monitoring workflow. By integrating post-quantum cryptography, privacy-preserving encryption, and integrity verification of software updates, the new processes strengthen trust across onboarding, medical data transmission, and long-term gateway maintenance. The following subsections describe how the system is extended in this pilot to address the challenges identified in the As-Is setup.

9.6.1 “To-Be” Reference Scenario #1: Onboarding and Device Authentication

In the To-Be scenario, the onboarding and authentication process of the Personal Health Gateway (PHG) is significantly strengthened to address the cryptographic limitations identified in the As-Is design. The plain HTTP-based registration flow used in the current system does not protect against eavesdropping or device impersonation, and relies solely on a shared MAC-secret pair for authentication. As the TelluCare deployment scales to potentially hundreds of thousands of patient homes, these weaknesses become increasingly critical, particularly in light of the emerging threat posed by quantum-capable adversaries that could compromise classical cryptographic schemes. To address this, To-Be Scenario #1 introduces the PQ SDK as a server-side component into the onboarding workflow, establishing a quantum-safe secure channel based on ML-KEM between the gateway and the backend services. The PQ SDK is a core component developed within the PiQASO project, and its integration into the TelluCare onboarding flow represents the application-specific deployment of this capability as a secure intermediary between the gateway and the Onboarding Server.

9.6.2 “To-Be” Reference Scenario #2: Trustworthy Transmission of Medical Data

While the current data transmission approach provides adequate protection at TelluCare’s existing scale, the growing deployment footprint and the evolving threat landscape - particularly the emergence of quantum-capable adversaries - make it increasingly important to introduce stronger encryption guarantees throughout the full communication chain. As the number of deployed gateways grows to potentially hundreds of thousands of patient homes, the consequences of a data breach become significantly more severe, both in terms of patient privacy and regulatory compliance. Furthermore, classical encryption schemes used to protect data in transit and at rest may become vulnerable to quantum attacks in the foreseeable future. To address these concerns, To-Be scenario #2 introduces end-to-end hybrid encryption of patient measurements combining Functional Encryption (FE) and Attribute-Based Encryption (ABE), computed collaboratively between the gateway and the PQC as a Service component through Multi-Party Computation (MPC). This ensures that data are encrypted before leaving the patient’s home and remain encrypted throughout transmission and storage, with decryption performed only when necessary for authorized clinical access.

9.6.3 “To-Be” Reference Scenario #3: Robust and Trustworthy Software Updates

Although the current SSH and Ansible-based update mechanism provides a reliable foundation for remote software maintenance, the reliance on classical RSA/ECDSA authentication and the absence of cryptographic signing of update packages leave the process vulnerable to authenticity attacks as the deployment scales. As TelluCare grows to larger fleets of gateways operating in patient homes across volatile and potentially adversarial network environments, it becomes critical to ensure that not only the integrity but also the authenticity of every software update can be formally verified at the gateway level. Furthermore, the long-term threat posed by quantum-capable adversaries means that classical signing schemes may no longer provide sufficient assurance over the operational lifetime of deployed devices. To address these concerns, To-Be Scenario #3 introduces post-quantum digital signatures for update package signing and verification, a quantum-safe communication channel for update delivery, and blockchain-based state anchoring to provide tamper-evident evidence of the update history of each gateway.

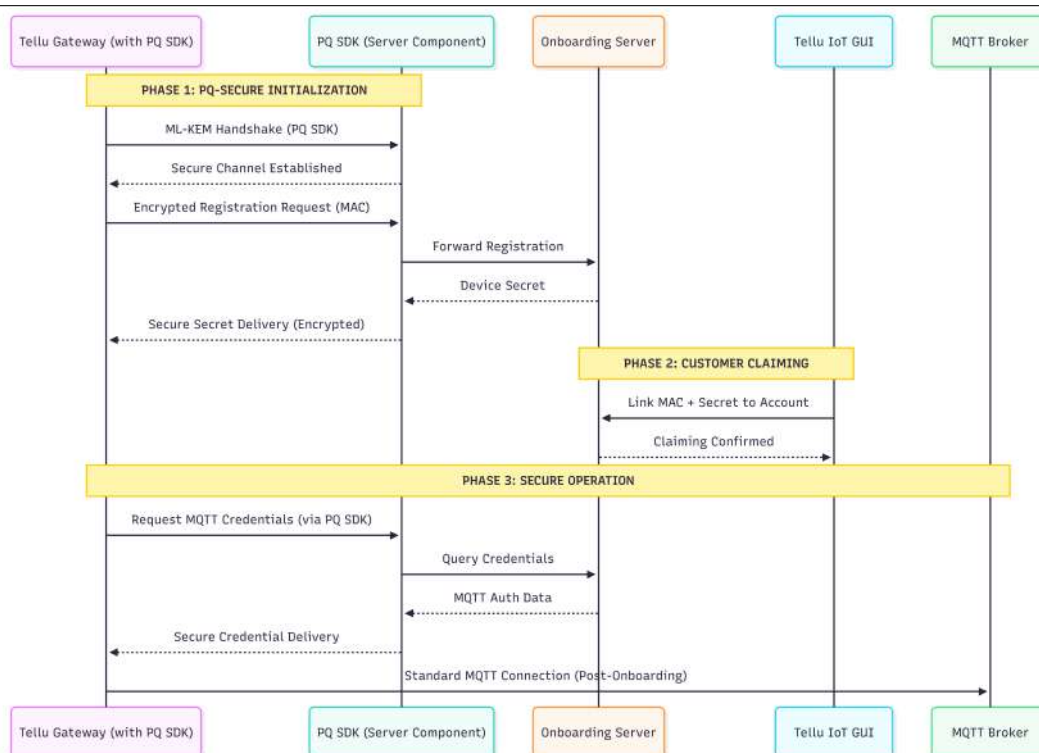


Figure 9.7: TO_BE onboarding of PHG to Tellu cloud sequence diagram

9.7 Reference Scenario User Stories and Metrics of Success

To evaluate the impact of PiQASO technologies in the TelluCare Remote Patient Monitoring pilot, a set of reference scenario user stories is defined. These user stories represent the key operational tasks performed by system administrators, security administrators, and authorised healthcare professionals responsible for onboarding gateways, protecting sensitive medical data, and maintaining the long-term trustworthiness of deployed devices. The user stories provide a structured basis for assessing whether the PiQASO-enabled To-Be scenarios achieve measurable security and operational improvements in an eHealth context.

9.7.1 User Story #1 (Reference Scenario #1): Secure and Scalable Gateway Onboarding

UC5.1.US1: Secure and Scalable Gateway Onboarding

As a System Administrator, I want to securely onboard new Personal Health Gateways into the TelluCare infrastructure using post-quantum secure authentication so that only legitimate and trusted devices can join the platform, and all authentication exchanges remain protected against both current and future quantum-capable adversaries.

The updated onboarding process follows the same three-phase structure as the As-Is scenario, with the key distinction that all sensitive communication is now conducted through a quantum-safe encrypted channel. The process is illustrated in 9.7.

Phase 1 – PQ-Secure Initialization

- **Step 1:** The Tellu Gateway (with PQ SDK client) initiates an ML-KEM handshake with the PQ SDK Server Component to establish a quantum-safe secure channel. Upon a successful handshake, an encrypted communication channel is established between the gateway and the PQ SDK.
- **Step 2:** Through the established secure channel, the gateway sends an encrypted registration request containing its MAC address to the PQ SDK Server Component.
- **Step 3:** The PQ SDK Server Component forwards the registration request to the Onboarding Server. The Onboarding Server generates a unique device secret associated with the gateway and returns it to the PQ SDK Server Component.
- **Step 4:** The PQ SDK Server Component delivers the device secret to the gateway in encrypted form over the secure channel. The gateway stores this secret locally for use in subsequent authentication steps. At this point, the device is registered as a legitimate gateway but is not yet associated with any customer account.

Phase 2 – Customer Claiming

- **Step 5:** The healthcare provider initiates account association by submitting the gateway's MAC address and the issued secret through the Tellu IoT GUI. This step links the physical gateway deployed in the patient's home to the correct customer account in the TelluCare platform.
- **Step 6:** The Onboarding Server confirms the claiming and notifies the Tellu IoT GUI that the account linking has been completed successfully.

Phase 3 – Secure Operation

- **Step 7:** On its next boot following successful account claiming, the gateway sends a request for MQTT credentials to the PQ SDK Server Component via the quantum-safe secure channel, authenticating itself using its stored MAC address and secret.
- **Step 8:** The PQ SDK Server Component forwards the credential query to the Onboarding Server, which retrieves and returns the corresponding MQTT authentication data.
- **Step 9:** The PQ SDK Server Component delivers the MQTT credentials securely to the gateway over the encrypted channel.
- **Step 10:** The gateway uses the received credentials to establish a standard MQTT connection to the MQTT Broker and begin streaming patient sensor data to the TelluCare cloud infrastructure, completing the onboarding workflow.

By introducing the PQ SDK as a quantum-safe intermediary, the To-Be onboarding process provides substantially stronger security guarantees than the As-Is design. All registration exchanges and credential provisioning are now protected against interception, replay, and impersonation attacks through the ML-KEM secure channel. The use of the PQ SDK, as a purpose-built PiQASO project component, ensures that the onboarding process remains cryptographically resilient against future quantum-enabled adversaries, addressing the long-term security concerns inherent in classical key exchange schemes. Together, these improvements establish a more trustworthy foundation for large-scale onboarding of PHG devices across distributed patient home environments.

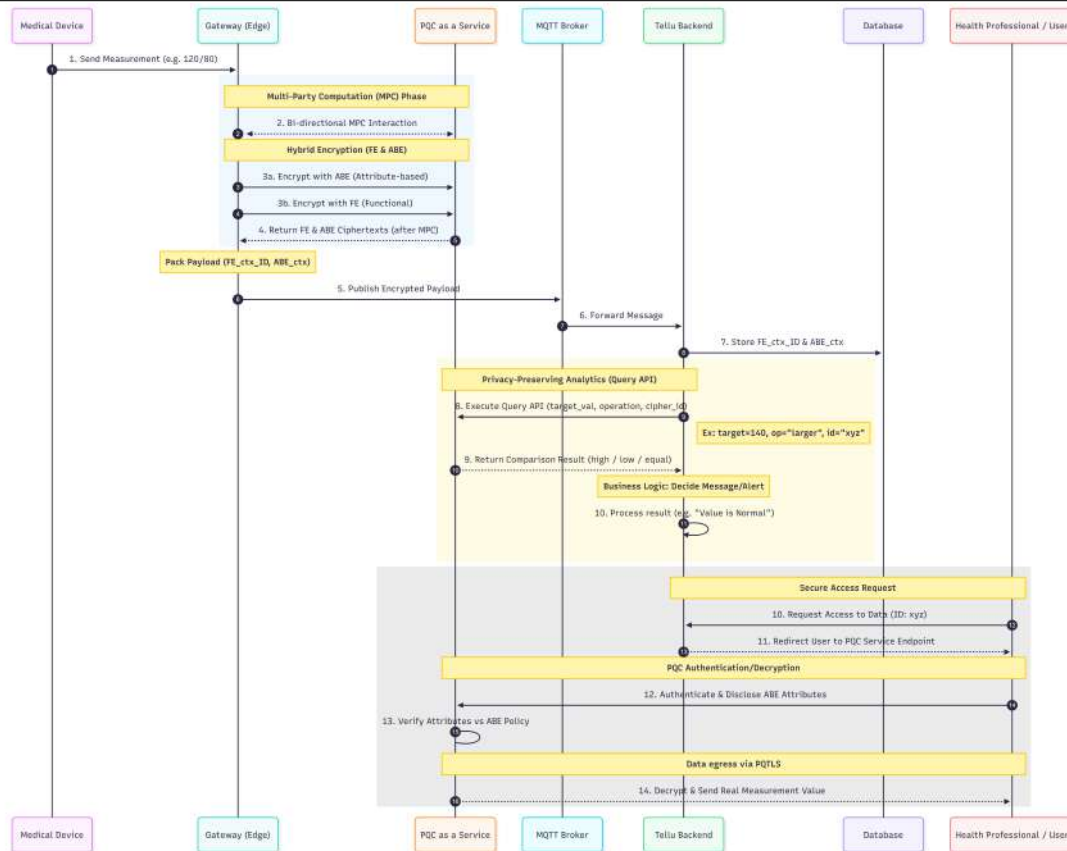


Figure 9.8: TO_BE sequence diagram for data transfer from PHG to Tellu services

9.7.2 User Story #2 (Reference Scenario #2): Confidential and Trustworthy Medical Data Transmission

UC5.1.US2: Confidential and Trustworthy Medical Data Transmission

As a Security Administrator, I want to ensure that patient measurement data remains confidential and encrypted throughout transmission, cloud-side processing, and storage so that sensitive medical information is never exposed in plaintext within backend services, and decryption is performed only when an authorised healthcare professional explicitly requires access to the real measurement value.

The updated data transmission process is illustrated in Figure 9.8.

- Measurement Acquisition.** The patient performs a health measurement using a supported medical device. The device transmits the recorded raw data (e.g. 120/80) to the Gateway over Bluetooth, as in the As-Is scenario.
- Multi-Party Computation (MPC) Phase.** Upon receiving the measurement, the Gateway initiates a bi-directional MPC interaction with the PQC as a Service component. The gateway and the service exchange messages collaboratively to jointly compute the encryption of the raw measurement value. Neither party performs the encryption independently; the cryptographic computation is distributed across both parties through the MPC protocol, ensuring that no single party has full knowledge of both the input and the encryption parameters at any point.
- Hybrid Encryption (FE and ABE).** As part of the MPC computation, two encryption schemes are applied simultaneously to the measurement value:

- a) The measurement is encrypted using Attribute-Based Encryption (ABE), producing an ABE ciphertext (ABE_ctx) that enforces an access policy, ensuring that only authorised healthcare personnel with the correct attributes can decrypt the data.
 - b) The measurement is simultaneously encrypted using Functional Encryption (FE), producing an FE ciphertext (FE_ctx) that enables privacy-preserving computation on the encrypted value without requiring decryption.
4. **Ciphertext Delivery.** Following the completion of the MPC phase, the PQC as a Service component returns both the FE ciphertext identifier (FE_ctx_ID) and the ABE ciphertext (ABE_ctx) to the Gateway.
5. **Payload Packaging and MQTT Publish.** The Gateway packages both ciphertexts into an encrypted MQTT payload and publishes it to the MQTT Broker. The transmitted payload contains no plaintext measurement values, ensuring that sensitive patient data is fully protected throughout transit.
6. **Message Forwarding.** The MQTT Broker forwards the encrypted message to the Tellu Backend for processing and storage.
7. **Encrypted Storage.** The Tellu Backend stores both the FE_ctx_ID and ABE_ctx directly in the Database without decryption. Patient measurements therefore remain encrypted at rest, eliminating the risk of plaintext data exposure through cloud-side attacks or insider threats.
8. **Privacy-Preserving Analytics.** The Tellu Backend performs analytical processing on the stored ciphertext by executing a query against the Query API exposed by the PQC as a Service component. The query is submitted with the parameters (target_val, operation, cipher_id), where target_val is the reference threshold value, operation specifies the comparison type (e.g., larger, smaller, or equal), and cipher_id identifies the specific ciphertext record in the database. The PQC as a Service evaluates the query directly on the encrypted data and returns a raw comparison result (e.g., higher, lower, or equal by a given margin) to the Tellu Backend, without exposing or decrypting the underlying plaintext measurement at any point.
9. **Business Logic Processing.** The Tellu Backend applies its own clinical business logic to interpret the raw comparison result returned by the Query API and determines the appropriate status to present to the healthcare professional (e.g., normal, warning, or alert). This separation ensures that the PQC as a Service component is only responsible for mathematical computation on encrypted data, while all clinical decision logic remains within the Tellu Backend.
10. **Secure Access Request.** When an authorised healthcare professional requires access to the real measurement value, they submit a request to the Tellu Backend for data access identified by a specific record ID. The Tellu Backend redirects the user to the PQC as a Service endpoint to perform authenticated decryption.
11. **PQC Authentication and Attribute Verification.** The healthcare professional authenticates to the PQC-as-a-Service by disclosing the required set of ABE attributes. The PQC-as-a-Service verifies the disclosed attributes against the ABE access policy associated with the requested ciphertext. If the attributes satisfy the policy, the service proceeds with decryption; otherwise, access is denied.
12. **Authorised Decryption and Data Egress.** Upon successful attribute verification, the PQCaaS decrypts the ABE ciphertext and transmits the real measurement value directly to the healthcare professional via a PQTLS-secured channel, ensuring that the decrypted data is protected against interception during delivery.

By introducing MPC-based hybrid encryption combining FE and ABE, the To-Be scenario ensures that patient measurements are never exposed in plaintext within any backend service or database. The use of the Query API enables the Tellu Backend to support clinical monitoring and threshold-based alerting workflows without ever requiring decryption of the underlying data. Decryption is strictly limited to authorised access scenarios, controlled through ABE attribute verification, and delivered securely via PQTLS. Together, these mechanisms substantially strengthen the confidentiality and privacy guarantees of the TelluCare platform, ensuring long-term compliance with strict healthcare confidentiality requirements as the deployment scales.

9.7.3 User Story #3 (Reference Scenario #3): Trusted Software Update Deployment

UC5.1.US3: Trusted Software Update Deployment

As a System Administrator, I want to deploy software updates to gateways with strong cryptographic verification of both the integrity and authenticity of every update package so that only authentic and authorised updates originating from a verified source can be installed, and the update history of each gateway is permanently recorded in a tamper-evident blockchain ledger to support auditability and compliance over the operational lifetime of the device.

The updated software update process is illustrated in 9.9.

1. **Key Management.** The Master Computer initialises post-quantum key management using the local PQ SDK, generating the cryptographic keys required for update signing based on ML-DSA-44 for digital signatures and ML-KEM-512 for secure key encapsulation. These keys form the cryptographic foundation for all subsequent update operations.
2. **Update Package Generation.** The Master Computer generates the software update package locally, assembling all necessary components and version metadata for the target gateway.
3. **Post-Quantum Signing.** The Master Computer signs the generated update package using ML-DSA-44 via the local PQ SDK. This signature cryptographically binds the update package to the authorised update server, ensuring that any tampering with the package after signing can be detected at the gateway.
4. **PQTLS Tunnel Establishment.** Before transferring the update, the Master Computer establishes a Post-Quantum TLS tunnel with the Tellu Gateway, providing a quantum-safe secure channel for the delivery of the signed update package and its associated metadata.
5. **Signed Update Delivery.** The Master Computer pushes the signed update package together with its metadata to the Tellu Gateway through the established PQTLS tunnel.
6. **Gateway Verification.** Upon receiving the update package, the gateway performs two verification steps using the PQ SDK:
 - a) **Authenticity Verification.** The gateway verifies the ML-DSA-44 digital signature of the received package, confirming that the update originates from an authorised source and has not been tampered with during distribution.
 - b) **Integrity Verification.** The gateway independently verifies the integrity of the update package contents, ensuring that no corruption or unauthorised modification has occurred during transfer.

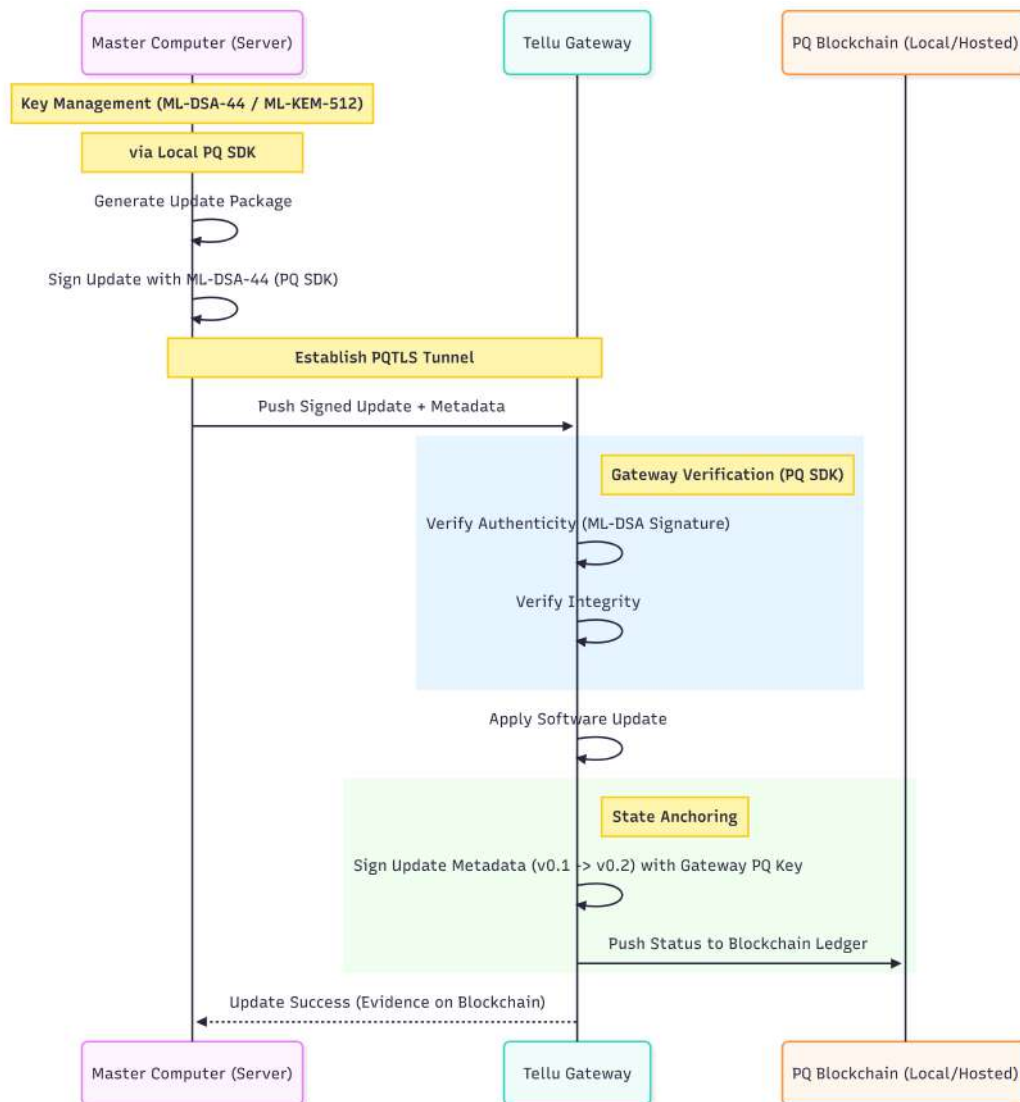


Figure 9.9: TO_BE sequence diagram for the update of gateways

If either verification step fails, the gateway rejects the update package and no installation is performed, preventing the deployment of unauthorised or compromised software.

7. **Update Installation.** If both verification steps pass successfully, the gateway applies the software update, installing the new software version on the device.
8. **State Anchoring.** Following successful installation, the gateway performs state anchoring to create a tamper-evident record of the update event:
 - a) The gateway signs the update metadata using its own gateway PQ key, producing a cryptographically authenticated record of the version transition.
 - b) The signed update metadata is pushed to the PQ Blockchain Ledger, where it is permanently recorded as an immutable entry in the gateway's update history.
9. **Confirmation.** The Master Computer receives a confirmation that the update has been successfully applied, accompanied by the blockchain evidence reference, providing verifiable proof that the correct and authorised software version has been installed on the gateway.

By introducing ML-DSA-44 post-quantum signatures for update package authentication, PQTLS for secure update delivery, and blockchain-based state anchoring for tamper-evident update records, the To-Be scenario substantially strengthens the trustworthiness of the software update process across the TelluCare gateway fleet. Gateways can now formally verify the authenticity and integrity of every update before installation, eliminating the risk of update-based compromise. The blockchain anchoring further ensures that the update history of each deployed gateway remains auditable and tamper-proof over its full operational lifetime, supporting long-term security and compliance as TelluCare scales its deployment to large and distributed patient home environments.

9.7.4 Metrics of Success

In Table 9.3, we provide an indicative set of application-specific Key Performance Indicators (KPIs) against which the various operations and components of PiQASO will be evaluated, as outlined in the User Stories. The table is organised into two parts: pilot-specific KPIs tailored to the TelluCare Remote Patient Monitoring use case, and project-agnostic KPIs that are defined across all PiQASO pilots. Note that the values provided for the KPIs are indicative and may be refined and revised as the project progresses.

Table 9.3: PiQASO Application-Specific KPIs for the TelluCare Remote Patient Monitoring Use Case

KPI	Description	Target / Acceptance Criteria
A.1 Secure Gateway Onboarding		
PQ-TLS onboarding latency	Time required for the Personal Health Gateway (PHG) to complete a PQ-TLS handshake and mutual authentication with the onboarding backend service before receiving operational credentials. Measured from initiation of the connection request to successful credential provisioning, excluding network latency.	< 120 ms (excluding network latency). Successful onboarding rate $\geq$ 99%.
A.2 Post-Quantum Key Generation		
Authentication key generation (PQ-TLS)	Time required to generate the post-quantum key pair used for PHG authentication and mutual authentication with the TelluCare backend during the onboarding process.	< 100 ms.

Functional Encryption (FE) key generation	Time required to generate the Functional Encryption master and functional keys used to encrypt patient measurements at the gateway before transmission to the TelluCare backend.	< 100 ms per key setup operation.
OTA update signing key generation	Time required to generate the post-quantum signing key pair used to sign and verify software update packages distributed to deployed PHGs.	< 100 ms.
Combined key generation time	Total time required to generate all three post-quantum key types in sequence: PQ-TLS authentication key, Functional Encryption key, and OTA update signing key.	Combined generation time < 300 ms.
A.3 Trustworthy Medical Data Transmission		
Functional Encryption processing latency	Time required to apply Functional Encryption to a patient measurement at the PHG before MQTT transmission to the TelluCare backend. FE ensures that sensitive measurement data is encrypted at the edge before leaving the patient's home environment, with decryption only possible by authorised healthcare personnel.	< 100 ms per measurement.
ABE storage protection latency	Time required to apply Attribute-Based Encryption to patient data at the backend before secure storage in the TelluCare cloud database, including access policy enforcement.	< 100 ms per record. Access policy enforcement success: 100%.
End-to-end cryptographic processing latency	Total time elapsed from patient measurement acquisition at the PHG to secure encrypted storage in the TelluCare backend, covering FE encryption at the edge, MQTT transmission, and ABE-based storage protection.	End-to-end latency < 5 s. Data integrity and authenticity verification success: 100%.
A.4 Trusted Software Update Deployment		
OTA update signature verification	Time required for the PHG to verify the post-quantum digital signature of an incoming software or firmware update package before installation, ensuring authenticity and integrity of the update.	Signature verification completed in < 600 ms. Detection of tampered or unauthorised updates: 100% of test cases.
A.5 Scalability and Hardware Acceleration		
Gateway scalability	Ability of a single PHG to simultaneously manage secure data collection and PQC-protected transmission for multiple concurrently connected medical devices (e.g., blood pressure monitor, pulse oximeter, weight scale) without degradation in cryptographic performance.	Support for ≥ 5 simultaneous medical devices with $\leq 10\%$ degradation in cryptographic processing latency per device.
FPGA hardware acceleration factor	Reduction in execution time of PQC operations (PQ-TLS handshake, FE encryption, and OTA signature verification) achieved through FPGA-based acceleration on the gateway, compared against equivalent software-only execution on a Raspberry Pi baseline. ABE operations are performed at the backend and are therefore excluded from this measurement.	At least $15\times$ reduction in PQC operation execution time relative to the software-only Raspberry Pi baseline, ensuring data freshness for medical measurements.

Chapter 10

PiQASO UC5.2 - Sector: Healthcare - Enhancing a Smart Ambulance System with PQC capabilities

The smart ambulance use case (UC5.2) in PiQASO entails the secure transmission of patient data during the transportation of patients by ambulance to a healthcare institution (hospital). In case of an emergency involving a patient needing transportation to a hospital, an Emergency Transportation Management (EMT) operator receives a call for medical assistance and has the responsibility for assigning an available medical transportation vehicle (e.g., ambulance) to the patient, as well as for finding a destination hospital with appropriate and available capacity.

During transportation, connected medical devices (CMDs) are used to monitor the patient's condition until the patient is safely admitted into the hospital. In addition, the paramedics in the ambulance collect medically relevant information about the patient's condition.

Currently, during the transportation phase, this information is mostly contained in the ambulance, and it is not shared with the destination hospital. In specific circumstances, the information may be sent by email, fax or internal applications to the destination hospital, but the patient data is not provided in real time to the medical team at the destination hospital.

The smart ambulance UC envisions the creation of a secure and connected smart ambulance system, allowing patient medical data to be sent in “near real-time” to a remote patient monitoring system (RPM) available at the point-of-care (i.e., the destination hospital) to medical teams. The patient information is thus instantly available to medical personnel having the appropriate permissions to access patient data (e.g., authorised hospital doctors and staff), facilitating the timely and efficient delivery of healthcare services upon arrival. An overview of the smart ambulance UC is depicted in Figure 10.1.

Within the scope of PiQASO, the following system functionalities are considered as part of UC5.2:

- The transmission of patient data collected by CMDs in the ambulance to the destination hospital;
- The access to the patient data by medical staff (e.g., doctors and nurses) at the destination hospital, according to their access control privileges as defined by the destination hospital;
- The network operation with varying characteristics, including throughput, reliability and availability.

From a cybersecurity perspective, this UC encompasses several challenges and needs:

- **End-to-end security:** The ambulance system operates over public networks (e.g., telecommunications infrastructure) but needs to securely transmit sensitive data (e.g., patient's health vitals

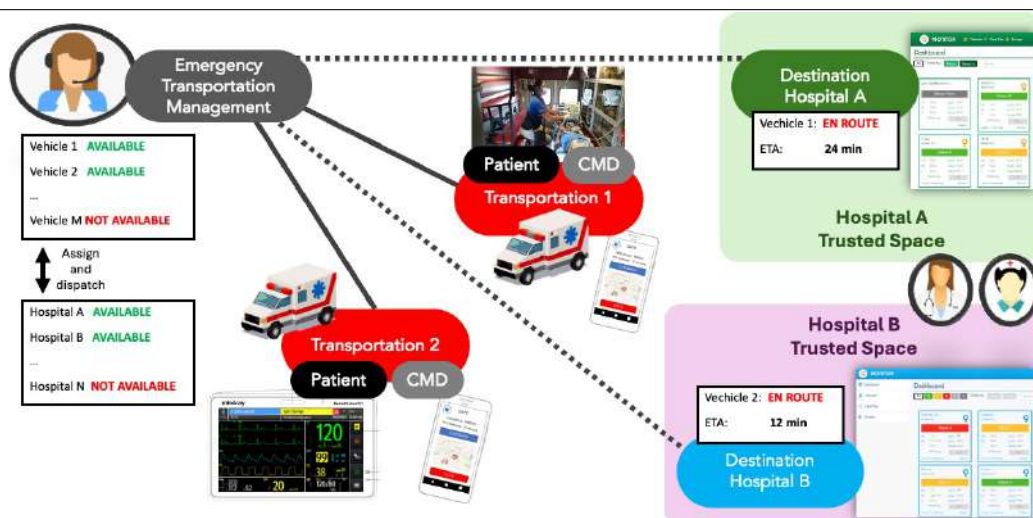


Figure 10.1: Smart Ambulance Scenario: Overall View

collected by the CMD) to recipient hospitals. However, public networks are a shared infrastructure with the general public, and malicious actors may exploit it by eavesdropping data communication packets. Security mechanisms - PQC-based - must be implemented to be ready against quantum-capable adversaries.

- **Dynamic and adaptable trust-building configuration between systems:** The ambulance system, including its onboard CMDs, operates as an independent system external to the destination hospital system. The destination hospital cannot set control policies on ambulance systems, thus trust relations need to be built “on-demand” (upon patient assignment). Hospitals consider their internal networks as part of their “trusted space”, while external systems, like the ambulance system, are outside this space. Smart ambulances need to provide the necessary trustworthiness guarantees for the establishment and maintenance of strong trust relationships with the destination hospitals.
- **Robust security framework in a mobile scenario:** The ambulance system is highly mobile and its connectivity relies on the public telecommunications infrastructure, which exhibits varying characteristics across regions and places. High dense population areas (e.g., cities) are typically well connected with broadband access, whereas remote areas (e.g., mountains) might be limited in bandwidth or have no connectivity at all. The implemented security framework needs to deliver high security features at all times, regardless of network performance.

10.1 UC Goals and Objectives

UC5.2 addresses the critical challenge of securely transmitting to the destination hospital the patient data collected by CMDs in an ambulance during transportation. The patient’s data is transmitted from the smart ambulance “on the move” to the destination hospital. Given the high mobility scenario, data transmission uses available public networks, accessible to the general public, including malicious actors. The evolution of quantum computers with the capability to break widely used cryptographic algorithms will eventually allow eavesdropping packets in public networks, thus posing significant risks to the transmission of sensitive health data, patient privacy and trust in digital healthcare.

This UC implements the PQC mechanisms developed in PiQASO in a smart ambulance system, fostering a system that is secure and ready against quantum-capable adversaries, even when operating over public networks and in highly dynamic settings.

The primary goal of this Use Case is to secure all transmitted patient data while ensuring a robust trust relation between ambulances and hospitals using PQC that benefit from PiQASO's functionalities.

The specific objectives are:

- **To ensure end-to-end security**, through the use of PQC-TLS to protect transmitted data against quantum-capable adversaries.
- **To enable trust-building configurations between ambulance and hospital systems** as a result of the PiQASO security framework supporting the establishment of secure and authenticated communication channels that facilitate the creation of strong trust relationships between ambulances and hospital systems.
- **To support different algorithm suites**, adapting and selecting the ones that match the requirements of a destination hospital IT system, ranging from legacy to the latest PQC algorithms.
- **To support highly mobile and dynamic scenarios**, using robust crypto enablers (e.g., ML-KEM and ML-DSA) optimised to operate on constrained and embedded devices.

10.2 PiQASO Functionalities

The smart ambulance UC showcases the integration of specific PQC functionalities developed in PiQASO to secure patient data transmission and storage between ambulance and hospital systems, considering a highly mobile scenario. The following table presents these functionalities and their specific application within the smart ambulance ecosystem, as well as the planned reporting period for their demonstration.

Table 10.1: PiQASO Functionalities to be evaluated for the Smart Ambulance UC

PiQASO Functionality	Description and Purpose in the Use Case	RP1	RP2
PQ-TLS	Establishes a secure communication channel between the ambulance and the hospital using quantum-resistant handshakes and session key encapsulation. Two variants will be used: PQC-as-a-service PQC-SDK (with local HW acceleration)	✓	✓
HW Acceleration	Uses FPGA-based hardware acceleration to offload PQC primitives, ensuring real-time performance at the edge.		✓

10.3 Components and Roles Considered

PARTICLE's Ambulance System involves connecting the physical CMDs in the ambulance to the PARTICLE gateway (GW) that links the CMDs to PARTICLE's cloud platform, where the data from the CMD is stored. Since the smart ambulance is a mobile environment, the connectivity is ensured using cellular connectivity via a mobile hotspot provided by an internet service provider (ISP).

The components present in the architectural setup of this use case are depicted in Figure 10.2 and can be summarised as follows:

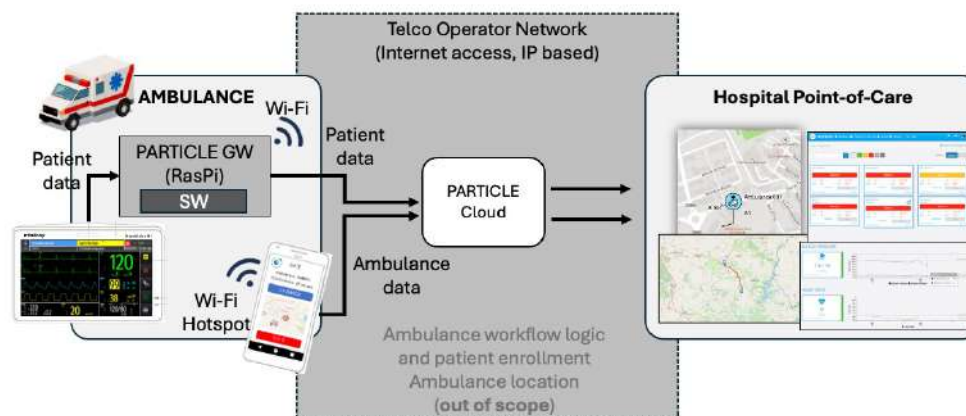


Figure 10.2: Ambulance UC Architecture: Components

- **PARTICLE Cloud Server (server node, MONITOR):** back-end and front-end services with API to receive, store and manage data from CMDs.
- **PARTICLE Gateway (proxy CMD):** Component deployed in the ambulance that provides connectivity between local CMDs and PARTICLE's MONITOR server. It will be used to run local PiQASO components.
- **Benevision N1 MD (medical device from third-party)[44]:** Medical device deployed in the ambulance that collects the patient's vitals data (blood pressure, heart rate, blood oxygenation and respiratory rate).
- **PARTICLE Ambulance App:** Support component implementing the Ambulance workflow process and providing real-time location information. The App is out-of-scope of PiQASO.

The following roles are identified:

- **Patient:** the person transported by the ambulance in need of medical attention.
- **Ambulance paramedic:** the medical professional monitoring the health condition of the patient during transportation.
- **Medical staff:** the staff at the hospital responsible for providing medical assistance when the patient arrives at the hospital.
- **IT administrator:** the staff responsible for operating and monitoring the functioning of the hospital's IT system, including identifying and responding to cyber incidents.

10.4 "As-Is" Scenario

In the "As-Is" scenario, the security of the communications system relies on the standard widely used Public Key Infrastructure (PKI). A PKI allows entities to securely communicate on insecure networks, thus being appropriate for the smart ambulance UC. Digital certificates are used to prove the validity of

its holder and the involved components perform operations using RESTful APIs with HTTPS and TLS. At the time of the writing, TLS latest version is v1.3 (RFC 8446), which supports several cryptographic algorithms such as Ephemeral Diffie-Hellman, AES-GCM, SHA-256, SHA-384, RSA-PSS. The smart ambulance system connects to PARTICLE servers that are digitally signed with “X9.62 ECDSA Signature with SHA-384” and use “TLS_AES_256_GCM_SHA384” for encryption.

PKI acts as the trust framework that enables TLS to function securely.

While TLS v1.3 is considered secure, its key exchange (ECDHE) and authentication (RSA/ECDSA) are both vulnerable to highly capable quantum computers, being possible to decrypt the data or impersonate a server.

The smart ambulance scenario involves entities that need to transmit sensitive data and provide access only to authorised entities. The scenario is highly mobile and data transmissions occur over public networks, thus strong security mechanisms (e.g., digital certificates and HTTPS using TLS v1.3) are implemented to ensure data confidentiality.

Device Authentication

PARTICLE GW follows a token-based authentication mechanism, as depicted in Figure 10.3. Its verification follows the establishment of a secure TLS tunnel as presented in the next diagram. The first part of the sequence diagram (steps 1 to 6) consists in establishing a TLS handshake using state-of-the-art techniques.

Once the TLS tunnel is established, the GW verifies its credentials by submitting its token to PARTICLE’s server for validation. The token is verified by PARTICLE’s backend component, returning the request with a success result. It is noted that these exchanges occur inside the TLS tunnel and are protected with strong encryption.

A critical element in the described flow consists in the transmission of the GW token. If this step is compromised, non-authorised third parties can get access to and perform operations in the system (e.g., send fake patient data).

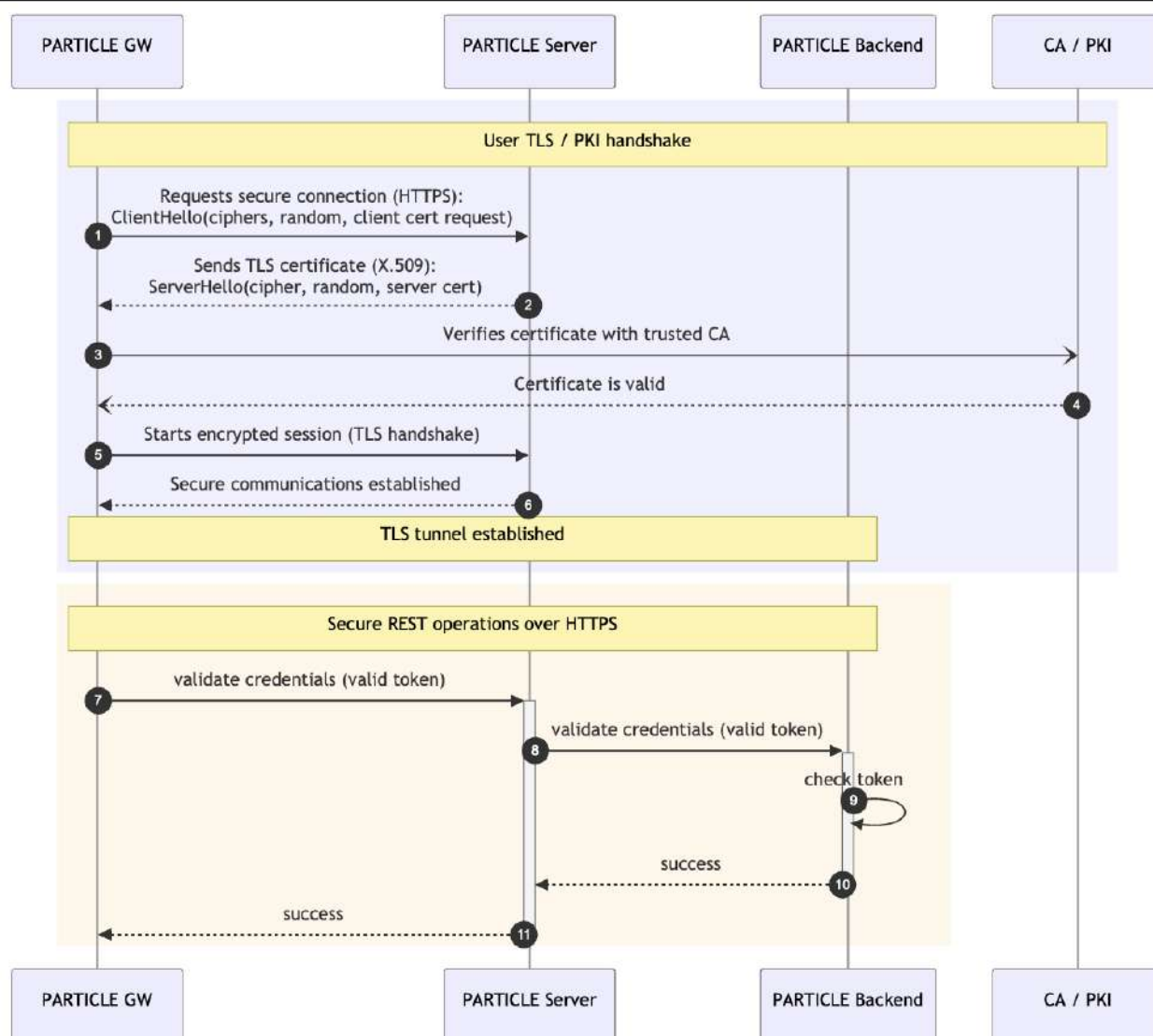


Figure 10.3: "as-is" Smart Ambulance: GW Authentication

User Authentication

Hospital doctors and nurses (i.e., users) use PARTICLE's MONITOR system to access the patient data transmitted by the smart ambulance system. For that, users need to authenticate in the MONITOR system using their username and password. The process is depicted in Figure 10.4.

Similarly to the GW authentication, the first part of the sequence diagram (steps 1 to 8) consists in establishing a TLS handshake between the MONITOR system and PARTICLE's server using state-of-the-art techniques. Once the TLS tunnel is established, the User is granted access to the system, where authentication details (i.e., username and password) can be inserted and submitted to PARTICLE's server. The credentials correctness is verified by PARTICLE's backend component, returning the request with a temporary token associated with the user. This token can be used in subsequent operations, reducing the risk of exposing username and password in every performed request. It is noted that these exchanges occur inside the TLS tunnel and are protected with strong encryption.

Critical elements in the described flow consist in the transmission of the user credentials (username and password) and the token. If these steps are compromised, non-authorised third parties can impersonate the user and gain access to patient data.

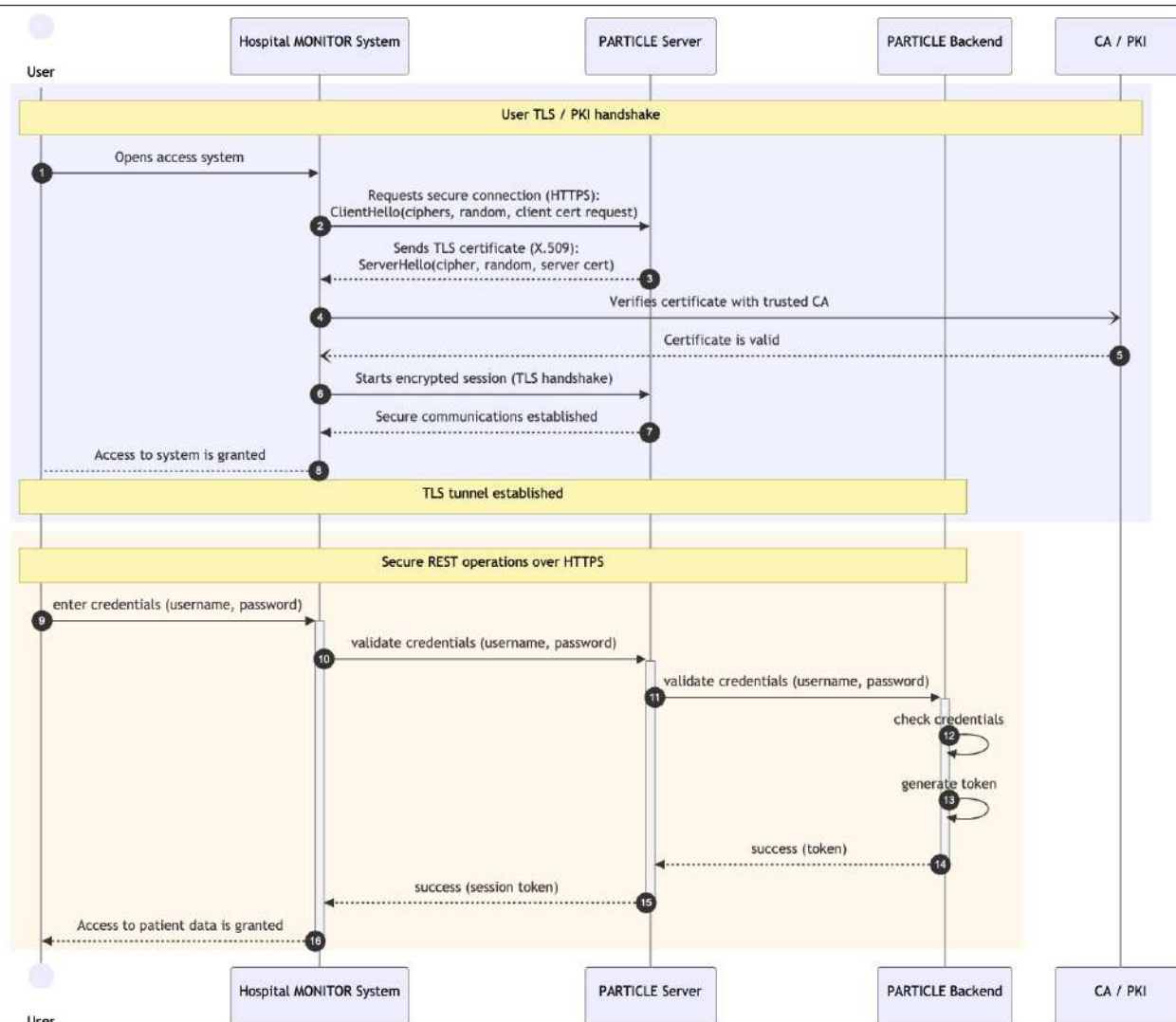


Figure 10.4: "as-is" Smart Ambulance: User Authentication Steps

Sending patient data

During ambulance transportation, the patient's vitals (e.g., blood pressure, heart rate, blood oxygenation, temperature, respiratory rate) are collected by CMDs and sent to PARTICLE's GW via wired LAN. PARTICLE's GW then sends the collected patient data to PARTICLE's server. The process involves several verification steps, as shown in 10.5.

The first part of the sequence diagram (steps 1 to 6) consists in establishing a TLS handshake between the GW and PARTICLE's server. Once the TLS tunnel is established, the GW sends patient data collected by the CMD to PARTICLE's server. The GW includes in this request its token. The request is forwarded to PARTICLE's device manager that verifies the device itself, verifies the GW token (using PARTICLE's backend component) and stores the patient data using PARTICLE's backend component. It is noted that these exchanges occur inside the TLS tunnel and are protected with strong encryption.

Critical elements in the described flow consist in the transmission of the GW token and patient data. If these steps are compromised, non-authorised third parties can perform operations in the system (e.g., send fake patient data) or gain access to transmitted patient data.

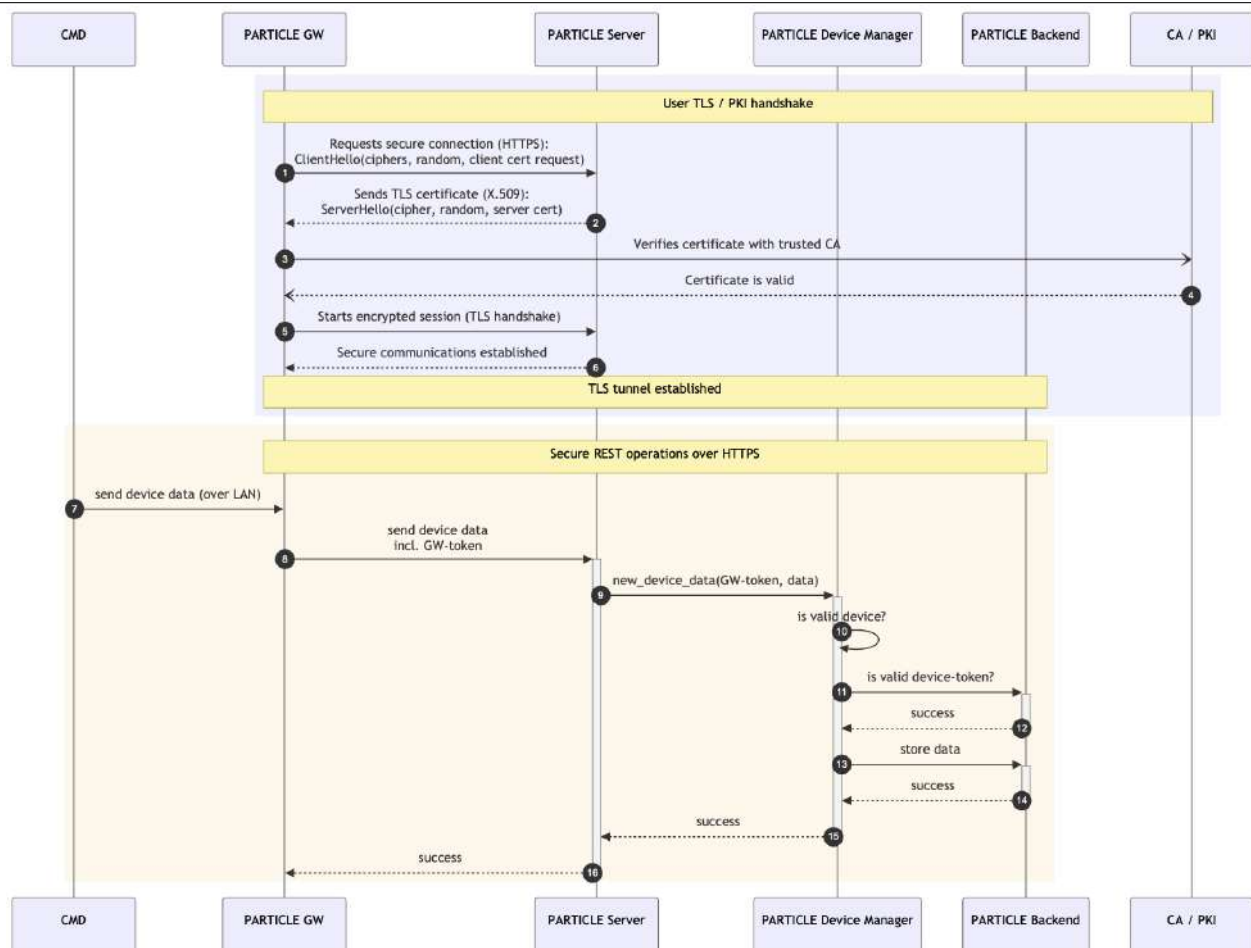


Figure 10.5: "As-is" Smart Ambulance: Sending patient data

10.5 "To-Be" Reference Scenario #1: Secure authentication using PQ-TLS via PQCaaS

This scenario brings quantum-resistant secure communications with the incorporation of PQ-TLS, using the PQCaaS adapter. It replaces the current TLS handshake process used for PARTICLE's GW authentication and User authentication in PARTICLE System.

Selected ML-KEM, ML-DSA, credential-management and session-establishment operations are submitted to PQCaaS over an authenticated (quantum-resistant) service channel.

Following the PQCaaS session establishment, the involved components (PARTICLE GW, PARTICLE server and PARTICLE MONITOR System at the hospital) exchange information and perform transactions within this secure channel.

This scenario involves replacing the existing schemes (e.g., ECDSA P-256 signatures) with PiQASO post-quantum signature schemes (e.g., ML-DSA) representing a straightforward migration path.

This configuration simplifies and minimizes the impact on the existing PARTICLE system, by relying on an external service for PQC session establishment, however, it also creates an additional dependency to PQCaaS and introduces network latencies associated with the session establishment. Moreover, considering the inherent mobile characteristic of this scenario - resulting in likely network disruption and/or

disconnection periods - it will be necessary to assess the PQCaaS effectiveness to functioning in these conditions.

10.6 "To-Be" Reference Scenario #2: Secure sending and accessing patient data using PQ-TLS via PQCaaS

Similarly to the previous one, this scenario incorporates PQ-TLS using the PQCaaS adapter, replacing the current TLS handshake process used for PARTICLE's GW when sending patient data to PARTICLE's server. Likewise, PARTICLE's MONITOR system at hospitals will access patient data, following a PQC session establishment with PARTICLE's server.

This scenario introduces the following specificities.

During transportation, patient data needs to be periodically transmitted (Patient health data is transmitted every 30 seconds or less) to PARTICLE's server and be *immediately* accessible to hospital users. PiQASO's required workflows need to efficiently perform required computations with low latencies (see acceptance criteria metrics in 3.7). Moreover, the unstable and dynamic network characteristics of the mobile environment will further stress the PiQASO's workflows, requiring robust and resilient properties to overcome situations like limited bandwidth and lack of network access. Note that the PQC-SDK with local accelerated HW (e.g., FPGA AMD / Xilinx EK-U1-ZCU104-G) will be tested to verify performance improvement and increased reliability.

In addition, the hospital ecosystem is diverse and may involve many different IT systems and configurations. As such, it is expected that different hospitals will support different algorithm suites, ranging from the latest PQC algorithms to legacy (non-PQC). As such, PiQASO crypto agility will need to be tested under such circumstances, as to ensure patient data delivery, using the best possible security mechanisms supported by the destination hospital.

The incorporation of PQ-TLS will allow establishing a secure TLS tunnel capable to protect sensitive data (e.g., user credentials, device token and patient data) against quantum threats. This also brings significant computational challenges especially for constraints devices, such as PARTICLE's GW, therefore these scenarios will provide insights on the feasibility to adopt PQ capabilities at the edge.

10.7 Reference Scenario User Stories and Metrics of Success

10.7.1 User Story #1 (Scenario #1): Secure User Authentication

UC5.2.US1: Secure Authentication Credential Exchange for Medical Doctors

As a Medical Doctor working at the hospital, I want to ensure my credentials (username and password) are kept secure through a post-quantum secure exchange when I authenticate in the hospital's system so that my authentication credentials remain protected against both current and future quantum-capable adversaries.

The used mechanisms for key encapsulation, key exchange and encryption (i.e., PQ-TLS) should be effective against quantum-capable adversaries, ensuring user credentials are not exposed to third-parties. Exposed credentials would enable unauthorised malicious actors to impersonate users, thus gaining illegitimate access to the system, perform operations and accessing patient's data.

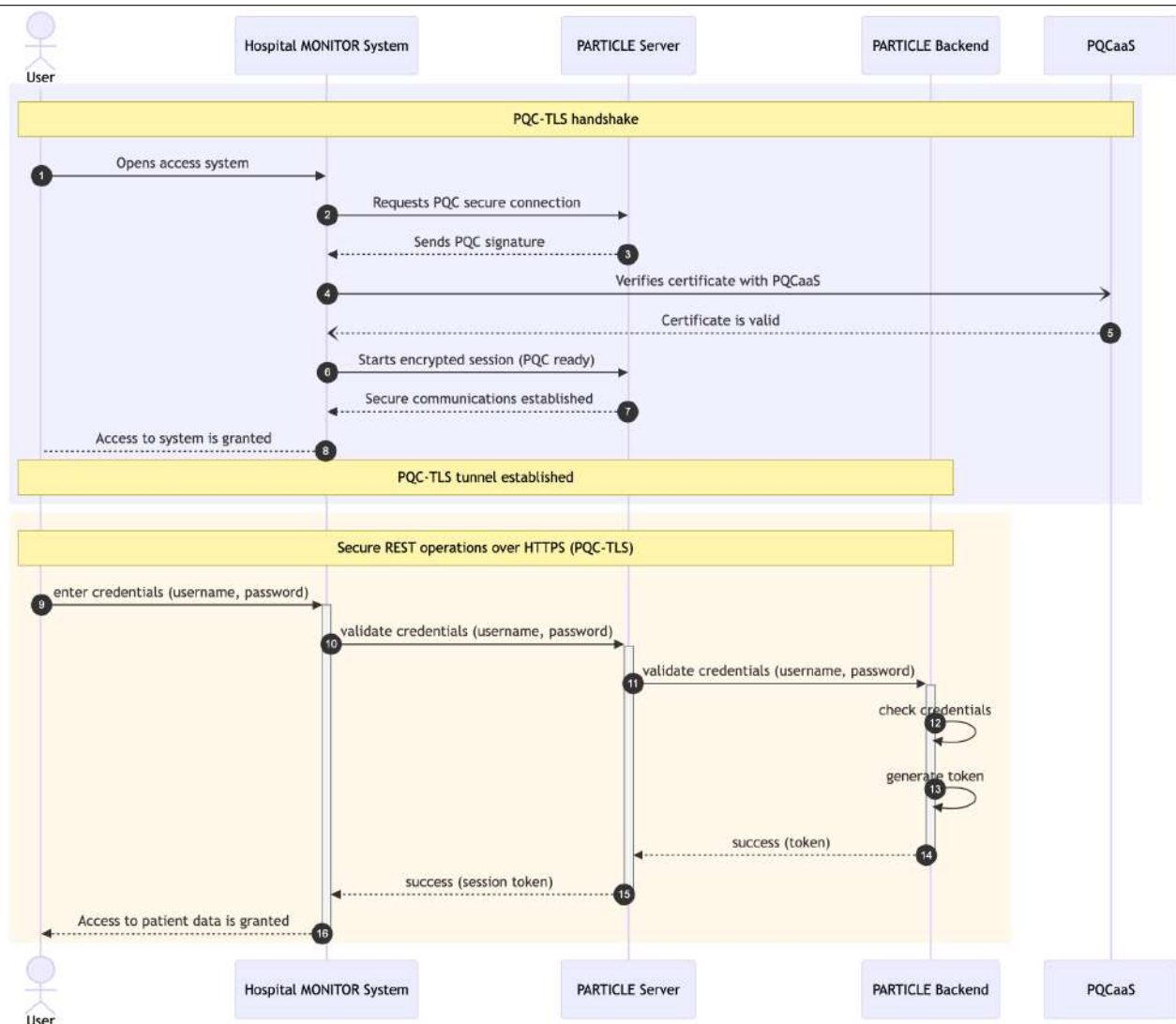


Figure 10.6: User Story 1: User Authentication with PQC-TLS

PQCaaS is used to establish a PQC-TLS tunnel between the hospital system and PARTICLE server, making the connection PQ secure (see figure 10.6). The approach results in a low impact on the existing system, since it is handled at TLS handshake level, leaving business-logic operations unchanged. However thanks to the established PQC-TLS, operations are now protected against PQ adversaries.

Workflow

PQ-TLS Session Establishment

1. User opens the MONITOR System
2. The MONITOR system requests a PQC secure connection with PARTICLE Server
3. PARTICLE Server replies with the PQC signature
4. The MONITOR system verifies PARTICLE Server's certificate with the PQCaaS (functioning as the trusted certificate authority)
5. PQCaaS replies indicating the certificate is valid
6. The MONITOR system starts a PQC encrypted session with PARTICLE Server

7. A secure communication is established
8. The User is given access to the MONITOR system

Once the PQ-TLS session is established, subsequent data exchanges are performed over a PQC-TLS secure tunnel.

9. The User inserts credentials (username and password) in the MONITOR system.
10. On user submit, the MONITOR system sends the user credentials to PARTICLE Server.
11. PARTICLE Server verifies the credentials with PARTICLE Backend.
12. PARTICLE Backend verifies that the credentials are valid.
13. PARTICLE Backend generates a temporary session token.
14. PARTICLE Backend returns the session token to PARTICLE server
15. PARTICLE server returns the session token to the MONITOR system
16. The MONITOR system grants access to the user.

10.7.2 User Story #2 (Scenario #1): Secure Device Authentication

UC5.2.US2: Secure Device Authentication Credential Exchange for Solution Providers

As a Solution Provider, I want to ensure my devices' credentials (device token) are kept secure through a post-quantum secure exchange so that only authorised devices can interact and perform operations on PARTICLE's servers.

As with the previous user story, the used mechanisms for key encapsulation, key exchange and encryption (i.e., PQ-TLS) should be effective against quantum-capable adversaries, ensuring GW device tokens are not exposed to third-parties.

Exposed credentials would enable unauthorised malicious actors to impersonate devices, thus gaining illegitimate access to the system, perform operations and sending fake patient's data.

PQCaaS is used to establish a PQC-TLS tunnel between PARTICLE GW and PARTICLE server, making the connection PQ secure (see figure 10.7). The approach results in a low impact on the existing system, since it is handled at TLS handshake level, leaving business-logic operations unchanged. However thanks to the established PQC-TLS, operations are now protected against PQ adversaries.

Workflow

PQ-TLS Session Establishment

1. PARTICLE GW requests a PQC secure connection with PARTICLE Server
2. PARTICLE Server replies with the PQC signature
3. PARTICLE GW verifies PARTICLE Server's certificate with the PQCaaS (functioning as the trusted certificate authority)
4. PQCaaS replies indicating the certificate is valid

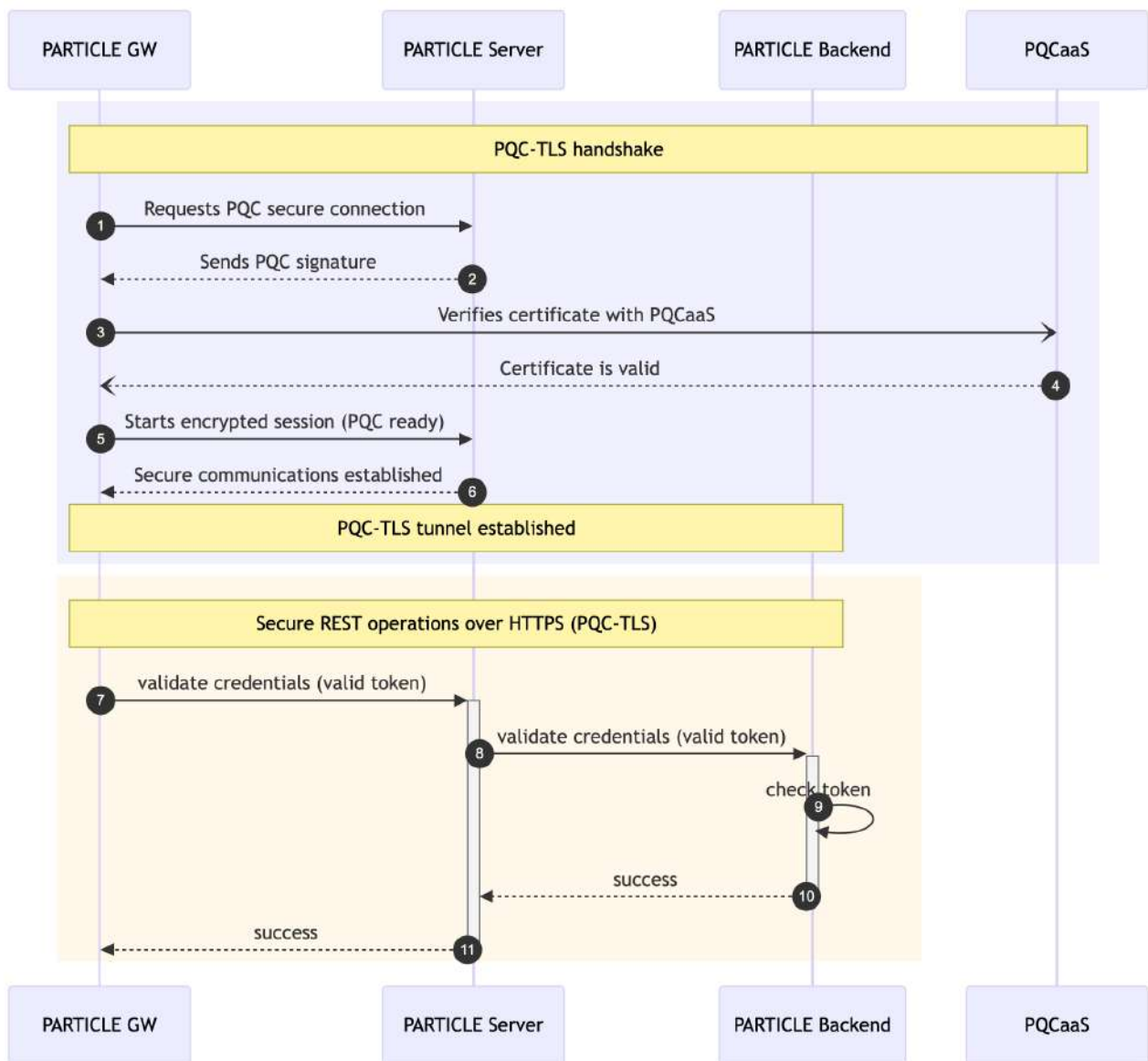


Figure 10.7: User Story 2: GW AUthentication with PQC-TLS

5. PARTICLE GW starts a PQC encrypted session with PARTICLE Server
6. A secure communication is established

Once the PQ-TLS session is established, subsequent data exchanges are performed over a PQC-TLS secure tunnel.

7. PARTICLE GW sends its credentials to PARTICLE Server.
8. PARTICLE Server verifies the credentials with PARTICLE Backend.
9. PARTICLE Backend verifies that the credentials are valid.
10. PARTICLE Backend returns success to PARTICLE server
11. PARTICLE server returns success to the PARTICLE GW (indicating it is authorized to perform actions)

10.7.3 User Story #3 (Scenario #2): Secure transmission of patient data

UC5.2.US3: Confidentiality of Patient Data Transmitted by PARTICLE's Gateway

As a Solution Provider, I want to ensure the confidentiality of patients' data transmitted by PARTICLE's Gateway using key encapsulation, key exchange, and encryption mechanisms (i.e., PQ-TLS) that are effective against quantum-capable adversaries so that transmitted patient data is not exposed to third parties, preventing malicious actors from causing harm to patients through the disclosure of their sensitive information.

The used mechanisms for key encapsulation, key exchange and encryption (i.e., PQ-TLS) should be effective against quantum-capable adversaries, ensuring transmitted patients' data are not exposed to third-parties. Malicious actors could cause harm to patients by disclosing their sensitive information.

Similarly to the previous user stories, this user story introduces PQaaS to establish a PQC-TLS tunnel between the gateway system and PARTICLE server, making the connection PQ secure (see figure 10.8). Once again, the approach results in a low impact on the existing system, since it is handled at TLS handshake level, with business-logic operations unchanged.

Workflow

PQ-TLS Session Establishment

1. PARTICLE GW requests a PQC secure connection with PARTICLE Server
2. PARTICLE Server replies with the PQC signature
3. PARTICLE GW verifies PARTICLE Server's certificate with the PQaaS (functioning as the trusted certificate authority)
4. PQaaS replies indicating the certificate is valid
5. PARTICLE GW starts a PQC encrypted session with PARTICLE Server
6. A secure communication is established

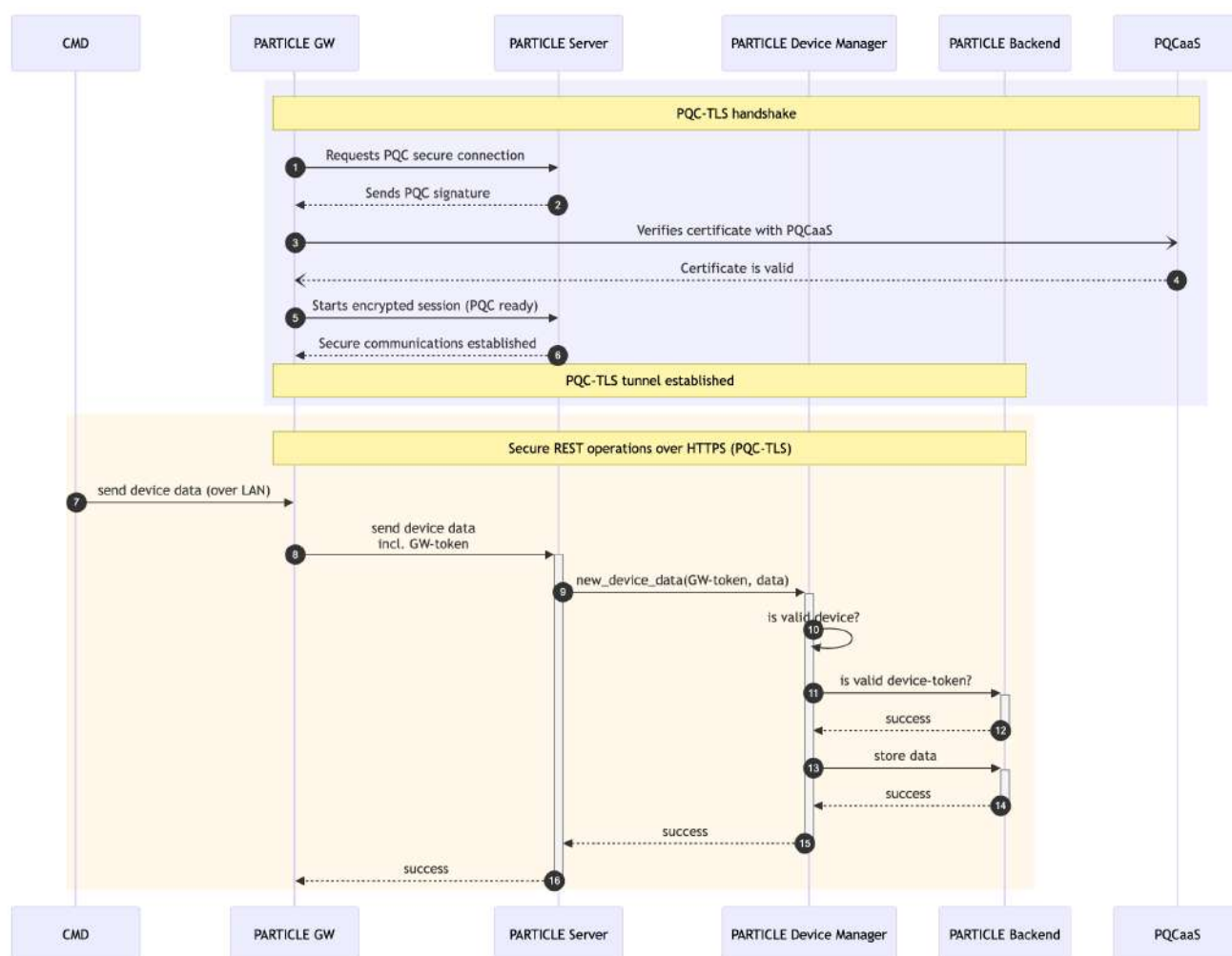


Figure 10.8: User Story 3: Secure transmission of patient data

Once the PQ-TLS session is established, subsequent data exchanges over the network (PARTICLE GW and PARTICLE Server) are performed over a PQC-TLS secure tunnel.

7. The CMD sends data to PARTICLE GW (over a wired LAN)
8. PARTICLE GW sends both its access token and CMD data to PARTICLE Server.
9. PARTICLE Server sends the new device data to PARTICLE Device Manager
10. PARTICLE Device Manager verifies that the device is valid
11. PARTICLE Device Manager requests a device token verification to PARTICLE Backend
12. PARTICLE Backend replies with success (i.e., the token is valid)
13. PARTICLE Device Manager sends the CMD to be stored by PARTICLE Backend
14. PARTICLE Backend returns success to PARTICLE Device Manager
15. PARTICLE Device Manager returns success to PARTICLE server
16. PARTICLE server returns success to the PARTICLE GW (indicating the data was successfully stored)

10.7.3.1 User Story #3.1 (Scenario #2): Querying on Patient Data

User Story 3.1: Attribute-Based Access and Query Control for Data Transmitted by PARTICLE's Gateway

As a Solution Provider, I want to ensure that only authenticated users with the required attributes can access data transmitted by PARTICLE's Gateway, or perform queries for operations on the encrypted data when decryption is not permitted so that sensitive data is only exposed to authorised parties, while still enabling legitimate analytical operations on encrypted data where the querying user is not permitted to perform decryption.

As an extension of User Story #3, this User Story refers to the case where an external authenticated user (e.g., doctor) wishes to obtain access or perform operations on encrypted data originating from the Smart Ambulance. In this regard, two modalities are provided: (i) Through **Attribute-Based Encryption (ABE)**, an external party who can verifiably demonstrate the possession of the required attributes can request and decrypt a set of encrypted data, or (ii) through **Functional Encryption (FE)**, so that the external party can perform queries on a set of encrypted data without decrypting them. For instance, a doctor may wish to check whether a specific value is present within the set of encrypted data, or if some patient vitals are above or below a specific threshold.

The sequence diagram of this Story is provided in Figure 10.9, and the corresponding action workflow is described in the following.

1. Phase 1: Data Encryption

- (a) The CMD in the ambulance sends device data to the PARTICLE Gateway over the local ambulance network (LAN).
- (b) The PARTICLE Gateway and the PQCaaS Server perform mutual **PQ-TLS authentication**, as detailed in Figure 3.1.

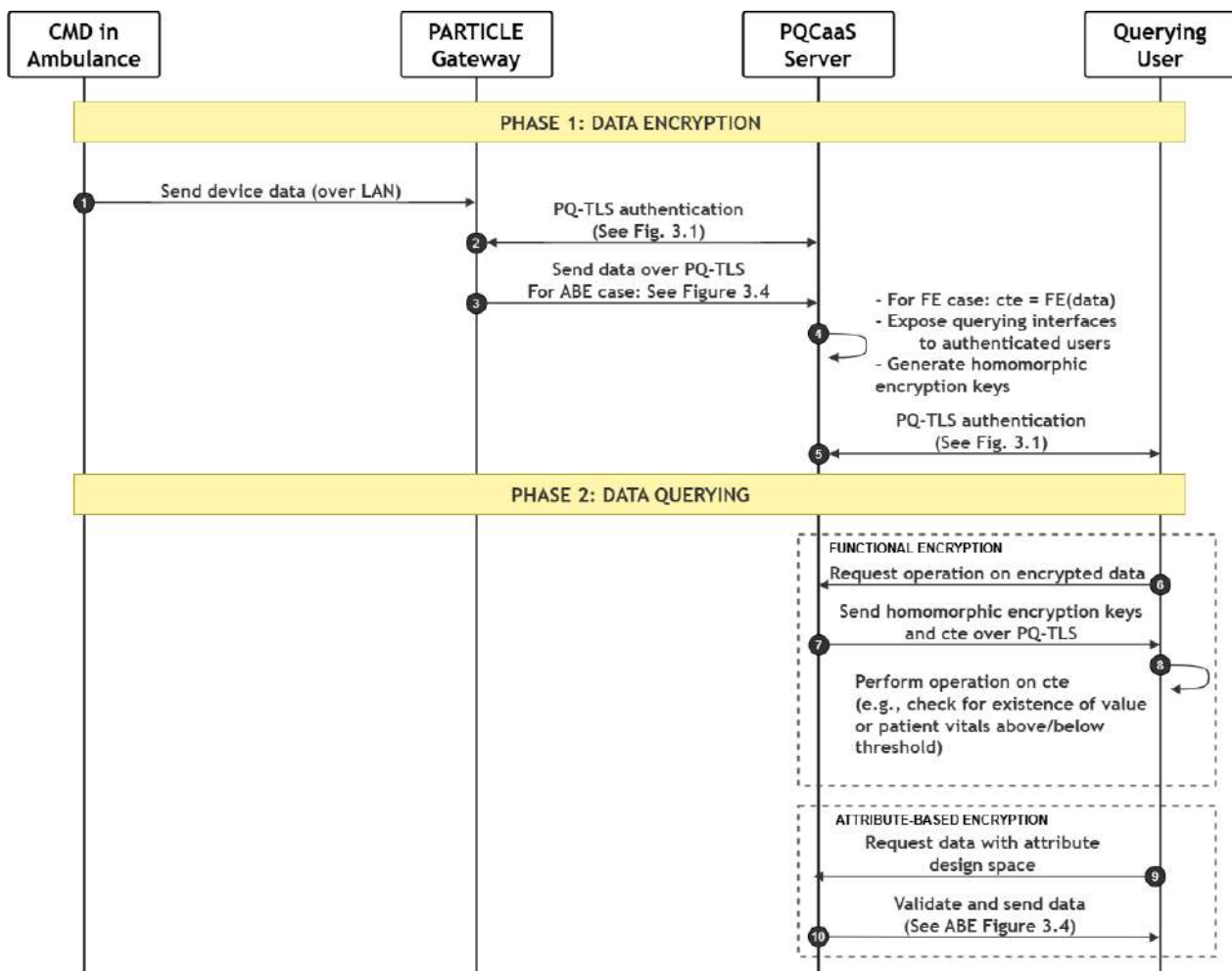


Figure 10.9: User Story 3.1: Querying on patient data with ABE or Functional Encryption

- (c) The PARTICLE Gateway sends the device data to the PQCaaS Server over the established PQ-TLS channel. For the ABE case, this exchange follows the procedure detailed in the common ABE sequence diagram in Figure 3.4.
- (d) The PQCaaS Server processes the incoming data: for the *Functional Encryption* case, it computes the ciphertext $cte = FE(data)$; it then exposes the corresponding querying interfaces to authenticated users and **generates the homomorphic encryption keys** required for subsequent querying.
- (e) The PQCaaS Server and the Querying User perform mutual **PQ-TLS authentication**, as detailed in Figure 3.1, establishing the secure channel over which Phase 2 queries will be carried out.

2. Phase 2: Data Querying

- (a) *Functional Encryption case*:
 - i. The Querying User requests an operation to be performed on the encrypted data from the PQCaaS Server.
 - ii. The PQCaaS Server sends the **homomorphic encryption keys** together with the ciphertext cte to the Querying User over the established PQ-TLS channel.
 - iii. The Querying User locally performs the requested operation on cte (e.g., checking for the existence of a value, or verifying whether patient vitals lie above or below a given threshold), using the homomorphic encryption keys. The technical details of this process are provided in D3.2 [56].
- (b) *Attribute-Based Encryption case*:
 - i. The Querying User requests data associated with a given attribute design space from the PQCaaS Server.
 - ii. The PQCaaS Server validates the user's attributes and sends the corresponding data, following the procedure detailed in the common ABE sequence diagram in Figure 3.4.

10.7.4 Metrics of Success

In Table 10.2, it is provided an indicative set of Key Performance Indicators (KPIs) against which the various operations and components of PiQASO in UC5.2 will be evaluated, as outlined in the User Stories, determining the conditions for success. The KPIs' values will be refined and revised as the project's quantum developments unfold.

Table 10.2: PiQASO KPIs for the Smart Ambulance UC

KPI	Description	Acceptance Criteria
Security		
Unauthorised access blocked	Effectiveness to block unauthorised access attempts	100%
Unauthorised data access blocked	Effectiveness to block unauthorised data access attempts	100%
Percentage of exposed data	Percentage of exposed data	In transit: 0% At-rest: 0%
Messages integrity and authenticity	End-to-end verification of integrity and authenticity of transmitted measurements	100% success
Performance		

Data transmission latency	Time from measurement collection on device to encrypted data delivery in PARTICLE Server	Using online PQaaS: $\leq 5s$ Using PQC with local HW ($\geq 5x$ time reduction compared with no local HW)
	CPU load	Average CPU load $\leq 25\%$ (peak 100% for less than 5 sec)
Network Robustness		
Recover from network failure	Capability to recover from network failure	100%
Reduced data throughput	Capability to support low data rates during runtime	kbits order
Scalability		
Number of ambulances	Maximum number of ambulances operating at the same time	Using PQaaS (simultaneous request): up to 10 Using PQC with local HW: up to 100
Reliability		
PQaaS	System availability of PiQASO PQaaS services	SLA 99%
Usability		
Percentage of PQ adoption	Percentage of system components involved in data exchange using PQ functionalities	100%
Integration Impact		
Integration impact	Integration of PQ in OS core functions, like PAM and openssl	Limited to OS configurations/updates
	Adaptations in solution source-code	Below 5% (measured modified lines of code against the total)

Chapter 11

PiQASO UC6.1 - Sector: Aerospace - Integration

11.1 Motivation and Challenges in Aerospace HIL Testing

11.1.1 Current Landscape: Hardware-in-the-Loop Validation in Aviation

The aerospace sector relies heavily on Hardware-in-the-Loop (HIL) testing during the critical integration phases of aircraft development. This methodology allows physical embedded systems - such as slat flap control computers, which alter the aerodynamic profile of the wing during takeoff and landing - to be rigorously validated within highly realistic, simulated flight environments before physical flight trials. Current integration testbenches require continuous, high-frequency interaction between the real System Under Test (SUT) hardware, virtual flight simulators calculating aerodynamic physics, and physical position sensors attached to actual motorized actuators.

These sophisticated testing environments rely on industrial communication standards like **OPCUA**, **Data Distribution Service (DDS)**, and **MQTT** to handle real-time control signals and telemetry. While these protocols excel at delivering low-latency performance - often requiring closed-loop responses in under 100ms to maintain aerodynamic stability - they were historically designed for isolated, trusted intranets. The current state-of-the-art leans heavily on perimeter-based network defenses, such as VLAN isolation and physical air-gapping, operating under the assumption that the test facility itself is a secure enclave.

However, as aerospace development becomes increasingly distributed across global supply chains, this perimeter-focused approach is showing its limitations. Testbenches are now frequently connected to broader corporate networks to facilitate remote diagnostics and collaborative engineering. This connectivity exposes critical middleware layers to significant vulnerabilities. Unauthenticated MQTT brokers or plaintext OPCUA and DDS exchanges can serve as primary attack vectors, potentially allowing malicious actors to inject anomalous sensor data or extract proprietary flight control logic.

11.1.2 The Convergence of Cybersecurity and Flight Safety

In aerospace HIL testing, the boundary between digital security and physical safety is non-existent. A compromise in the testing phase directly undermines the certification and safety guarantees of the final aircraft. Unlike standard IT environments where a breach results in data loss, tampering within a flight control testbench can lead to catastrophic physical outcomes once the hardware is deployed.

- **Validation Integrity:** An adversary exploiting unencrypted control protocols could subtly manipulate slat extension commands or falsify position sensor feedback. If a slat flap computer's response to tampered data is recorded as "successful validation," a flawed control algorithm could be approved for actual flight, posing severe risks to passenger safety.
- **Operational Disruption:** Denial of Service (DoS) attacks on the testing middleware can halt complex, multi-day simulation runs. For aerospace manufacturers operating on strict certification timelines, these disruptions translate into massive financial and scheduling penalties.
- **Intellectual Property (IP) Exfiltration:** Aerospace control algorithms represent decades of costly research and development. Unencrypted telemetry, test parameters, and system logs transmitted across the testing network are highly susceptible to espionage, exposing core competitive advantages to nation-state actors or industry competitors.
- **Operating-Envelope Profiling:** Even without altering a single packet, an adversary who passively records the plaintext command and feedback streams can reconstruct the slat actuator's complete operating envelope – its nominal range, its dynamic response, and, most sensitively, the exact thresholds at which the mechanism stalls, saturates, or fails. These operating bounds are simultaneously safety-critical intellectual property and a blueprint for a later, precisely targeted attack on the fielded aircraft. They must therefore never traverse the network in clear text: confidentiality of the sensor and command link is not a convenience but a primary safety control.

11.1.3 The Transition from Classical to Post-Quantum Cryptography in Aerospace

The aviation industry is uniquely exposed to the impending threat of quantum computing due to the extraordinarily long lifecycles of commercial and military aircraft, which often remain in service for 30 to 50 years. Consequently, the sector is heavily targeted by "harvest now, decrypt later" strategies, where adversaries intercept and store proprietary aerospace communications today, waiting for quantum hardware to mature.

Despite recent standardization efforts by NIST regarding Post-Quantum Cryptography (PQC), the aerospace testing community faces distinct, domain-specific hurdles in adopting these new cryptographic primitives:

- **Ultra-Low Latency Constraints:** Slat flap control systems require deterministic execution, with control response times strictly bounded under 100ms. The larger key sizes and computational overhead inherent in PQC algorithms threaten to violate these real-time constraints, potentially invalidating the HIL simulation dynamics.
- **Legacy Protocol Integration:** Existing aerospace testbenches rely on established protocols (OPCUA, DDS) that lack native support for hybrid or post-quantum cryptographic agility.
- **Distributed Complexity:** Modern aerospace testing is rarely centralized. Multi-node HIL systems span different geographic facilities, requiring secure, quantum-resistant authentication across wide area networks without degrading simulation synchronicity.

11.1.4 Motivation for PiQASO's Role in Aerospace Testing

PiQASO's contribution to aerospace HIL testing can be read along three axes. The **process** is the validation campaign itself, which exercises the System Under Test against a simulated aircraft until its operating envelope is fully characterised. That process produces and consumes **knowledge** – the control

laws, the sensor operating bounds, and the precise thresholds at which the mechanism fails – which is exactly the intellectual property an adversary most wants to read or corrupt. PiQASO supplies the **tools** that keep this knowledge confidential and authentic, in motion and at rest, without breaking the real-time process. The recurring design question across all three is one of bounds: the operating bounds that must stay secret, the latency bound that post-quantum protection must respect, and the hardware-acceleration ceiling that governs how much cryptography fits inside a control cycle.

The intersection of stringent real-time requirements, high-stakes intellectual property, and long-term quantum vulnerabilities creates a critical need for **validated, quantum-safe architectures in aerospace testbenches**. The PiQASO project addresses these specific challenges by:

- **PQC-Aware Control Protocols:** Demonstrating the integration of post-quantum functionalities (PQ-TLS, PQ Signatures) into MQTT, DDS, and OPCUA exchanges, bridging the gap between rigorous cryptographic security and sub-100ms timing constraints.
- **Hardware-Accelerated Proxies:** Leveraging **FPGA-based Trusted Computing Bases (QR-TCB)** as transparent network proxies. This offloads heavy PQC operations, ensuring cryptographic processing does not interfere with the SUT's simulated flight dynamics or trigger the need for expensive hardware recertification.
- **Distributed Authentication Frameworks:** Providing SDKs that enable secure, quantum-resistant identity management for SUTs and virtual simulators across geographically separated testing nodes.
- **IP Protection:** Implementing advanced encryption mechanisms for data-at-rest, ensuring that recorded test logs and proprietary algorithms remain cryptographically sealed against future quantum decryption.

11.2 Use Case Description, Goals, and Objectives

This Use Case focuses on the practical integration of post-quantum cryptography within a highly dynamic **Hardware-in-the-Loop testbench for slat flap control computers**. A defining feature of the BIBA demonstrator is **virtual testing (sim-in-the-loop)**: the same controller can be driven against either the physical actuator or a faithful software model of the motor over the identical, PQC-secured network link, with a one-click switch between the two and no change to the control software (detailed in Section 11.5.2).

In a standard simulation cycle, a high-performance Virtual Testing Computer generates dynamic flight conditions (e.g., altitude changes, turbulence) and sends control signals to the embedded SUT. Simultaneously, physical position sensors stream high-frequency feedback regarding the mechanical state of the slat actuators. The SUT processes this data, calculating necessary adjustments and firing electrical motor commands to achieve the desired aerodynamic configuration. This entire loop must execute deterministically. The arrival of quantum capabilities threatens to compromise the classical encryption protecting these loops, allowing adversaries to silently manipulate test variables or exfiltrate the proprietary physics models governing the aircraft's behavior.

The primary goal of this Use Case is to **secure all HIL testing communications and stored telemetry using post-quantum cryptography (PQC)** via PiQASO's framework, without violating the strict timing budgets of the aerospace domain.

The specific objectives are:

- Ensure **end-to-end confidentiality, authenticity, and integrity** of real-time HIL control signals (OPCUA/MQTT/DDS) against quantum-capable adversaries.

- Provide **PQC-secured authentication** for the onboarding of new System Under Test (SUT) hardware and remote virtual testing nodes.
- Maintain **real-time execution**, proving that PQC overhead can be managed to keep control response times strictly under 100ms.
- Enforce **fine-grained authorization** and **intellectual property protection** for proprietary flight control algorithms and recorded simulation data.
- Benchmark the latency, throughput, and resource overhead of PQC primitives directly within an active aerospace control loop.

11.3 PiQASO Functionalities

In alignment with the project objectives, the Aerospace Use Case will demonstrate the integration of specific PQC functionalities to protect HIL operations. The following table outlines these tools and their phased rollout.

Table 11.1: PiQASO Functionalities to be evaluated for UC6.1 (Aerospace Integration)

PiQASO Functionality	Description and Purpose in the Use Case	RP1	RP2
PQ Signatures	Used by the Virtual Testing Computer to sign motor commands, ensuring control signals are authentic and untampered.	✓	
PQ KEM	Encrypts high-frequency control payloads, protecting the confidentiality of the simulated flight parameters.	✓	
PQ-TLS	Establishes a secure, quantum-resistant communication channel between distributed HIL nodes and the central testbench.	✓	✓
QR-TCB & HW Accel.	Utilizes FPGA-based hardware inline proxies to offload PQC primitives from the SUT, preserving the < 100ms real-time latency requirement.	✓	✓
AKE	Authenticated Key Exchange for secure, ephemeral session establishment between the SUT and physical position sensors.		✓
UPKE	Direct encryption of simulation telemetry before transmission, protecting IP during transit.		✓
ABE	Attribute-Based Encryption applied to test logs, ensuring proprietary algorithms are accessible only to authorized engineers based on their policy roles.		✓

11.3.1 Rationale for the Phased Implementation

The deployment is strategically divided to mitigate integration risks. The **first reporting period (RP1)** focuses purely on securing the active, real-time control loop. By deploying PQ-TLS and hardware-accelerated signatures on the MQTT/OPCUA/DDS channels, the project will prove that quantum security can coexist with aerospace timing constraints. The **second reporting period (RP2)** shifts focus to intellectual property and distributed complexity, introducing UPKE and ABE to secure data-at-rest (test logs) and enforce role-based access for remote engineering teams.

Concretely, the three To-Be reference scenarios map onto the reporting periods as follows: Reference Scenario #1 (Quantum-Safe Real-Time Control) and Reference Scenario #3 (Hardware-Accelerated HIL Control) are addressed in **RP1**, as both concern the live, latency-bounded control loop; Reference Scenario #2 (Secure Distributed HIL with IP Protection, based on ABE/RBAC) is **deferred to RP2** and is not in scope for the September milestone.

11.4 “As-is” Scenario

The current operational architecture for the HIL testbench relies on a legacy, network-centric security model. The operation process prior to PQC integration is depicted in Figure 11.1.

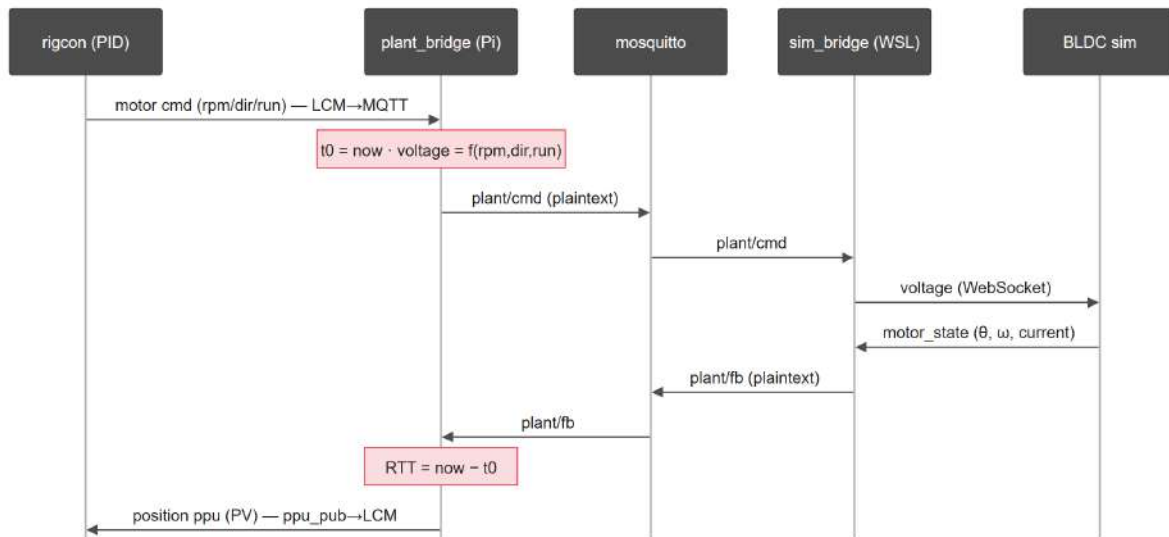


Figure 11.1: Operation process of the HIL rig before PiQASO: one control cycle with command and feedback exchanged in plaintext over the MQTT broker.

Scenario Narrative

The “As-is” architecture revolves around a core Virtual Testing Computer that orchestrates the simulation environment, pushing data to the Slat Flap Control Computer (SUT). Currently, the high-frequency exchange of position sensor data and motor control commands occurs via protocols like DDS and MQTT in **plaintext**. While basic network segmentation (VLANs) separates the testbench from the corporate network, application-layer cryptographic protection is entirely absent.

For multi-node testing, standard VPNs are occasionally utilized, but they operate on a “flat” trust model. Once a device or user is authenticated to the testing VLAN, they generally have unrestricted access to the middleware broker. This lack of granular authorization means a compromised low-level sensor node could theoretically be used to eavesdrop on the entire simulation suite. Furthermore, highly sensitive test results and proprietary algorithm parameters are written to standard network-attached storage using conventional classical encryption (or none at all), leaving the facility’s most valuable intellectual property exposed to “harvest now, decrypt later” strategies.

Summary of Key Vulnerabilities

- **Communication:** OPCUA, DDS, and MQTT traffic is exchanged without payload encryption, making control loops vulnerable to **interception, manipulation, and replay attacks**.
- **Networking:** The VLAN trust model is flat; there is no mutual authentication between individual SUTs and the virtual testing infrastructure.
- **Data Storage:** Test results and proprietary algorithms are stored with classical methods vulnerable to quantum computing threats.
- **Performance vs. Security:** Existing classical security measures (like standard TLS) are frequently disabled by test engineers to avoid latency spikes, leaving the system entirely exposed.

11.5 Components and Roles Considered

The ecosystem of the aerospace HIL testbench consists of the logical components presented in the following subsections.

11.5.1 System Architecture: Before vs After PQC

Figures 11.2 and 11.3 give the component-level architecture of the BIBA demonstrator before and after PiQASO integration. The control endpoint runs the slat-flap controller (`rigcon`); a lightweight bridge maps its internal signals onto MQTT topics; a broker carries the publish/subscribe traffic; and the plant – physical or simulated – closes the loop (see Section 11.5.2).

Before integration (Figure 11.2), command and feedback messages cross the broker in plaintext. After integration (Figure 11.3), a post-quantum framing daemon (`pq_frame`) is inserted at each endpoint: it performs the key agreement, then encrypts and authenticates every payload, publishing only ciphertext on the secured (`sec/*`) topics. Crucially, the broker is unchanged – it relays opaque ciphertext and never holds a key – so the security layer wraps the existing middleware rather than replacing it.

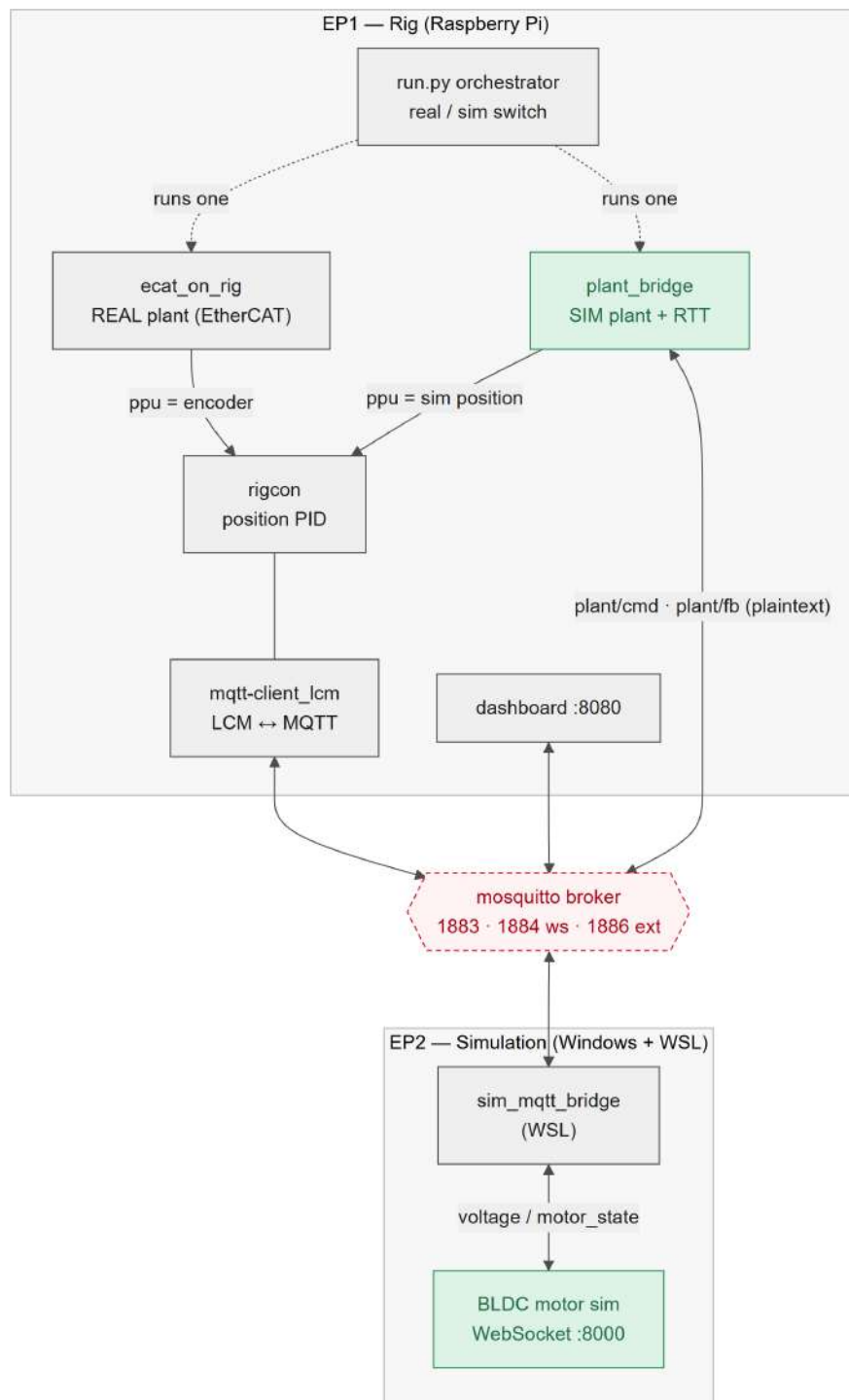


Figure 11.2: Demonstrator architecture before PiQASO: command and feedback cross the broker in plaintext.

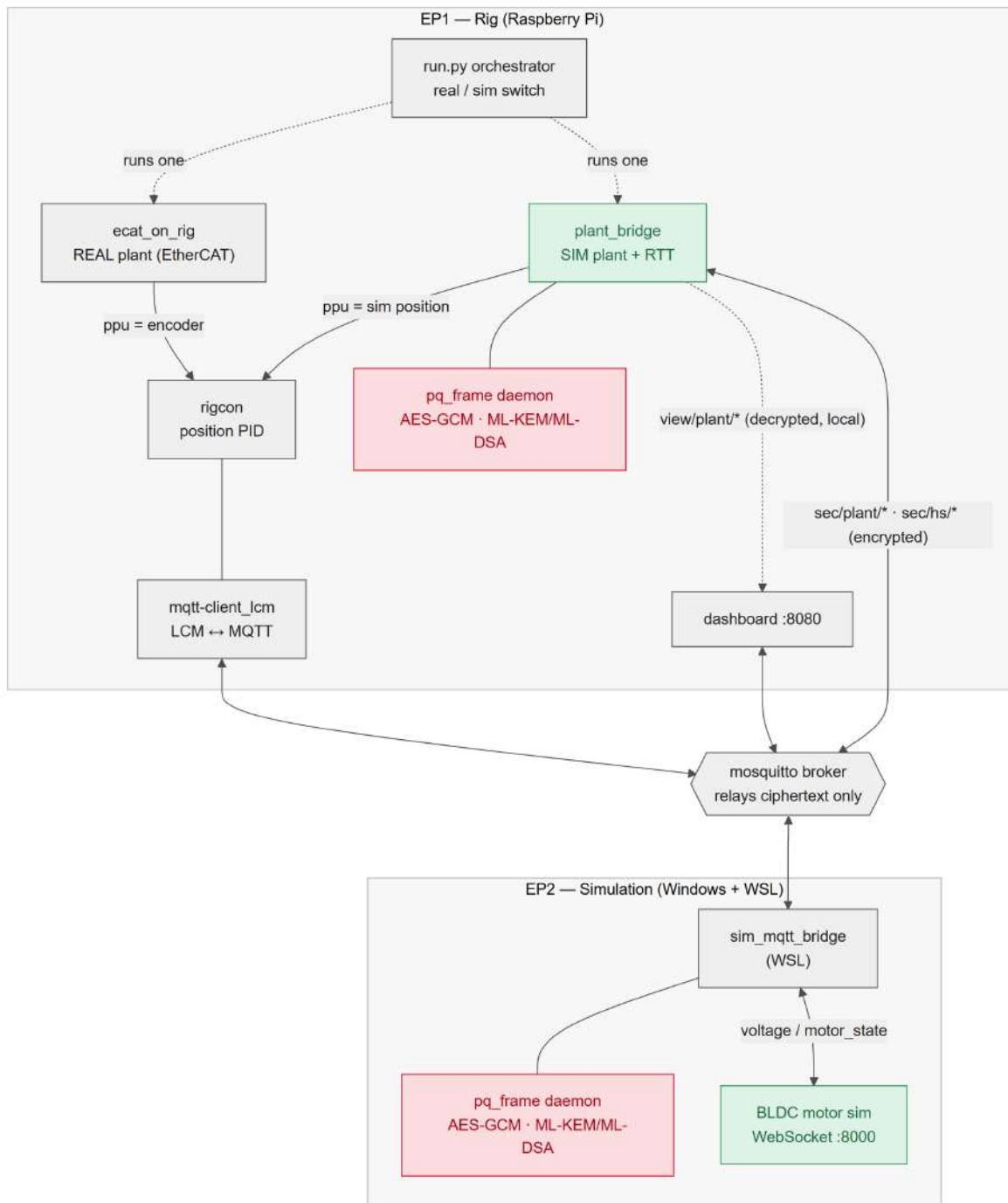


Figure 11.3: Demonstrator architecture after PiQASO: a `pq_frame` daemon at each endpoint secures the payloads; the broker relays ciphertext only.

11.5.2 Virtual Testing – Sim-in-the-Loop

A distinguishing capability of the BIBA demonstrator is **virtual testing**: the controller can be exercised against a faithful software model of the motor in place of the physical actuator, over the same PQC-secured network link. An orchestrator selects exactly one plant for any given run – the real actuator or the simulation – so the position feedback always arrives on the identical topic and in the identical format, whether it originates from the physical encoder or from the model. The controller (`rigcon`) is therefore unchanged and cannot tell the two apart, and the operator can switch between them live, without restarting the control software.

This matters for three reasons. First, it lets the secured control loop – and the post-quantum handshake that protects it – be developed, validated, and demonstrated at any time, without occupying or risking the physical rig. Second, it allows edge cases (disconnects, stalls, out-of-range commands) to be injected safely, with no hardware at risk. Third, because the cryptographic path is identical in both modes, performance and security results obtained in simulation transfer directly to the physical setup.

11.5.3 Components, Entities and Roles

System Entities and Roles

- **System Under Test (SUT) - Slat Flap Control Computer:** The real-time embedded hardware being validated. It operates on specialized aerospace-grade constraints and executes the core flight control logic. Because it must reflect flight-ready hardware, its internal firmware cannot be easily altered.
- **Virtual Testing Computer:** A high-performance workstation (e.g., Real-time Linux OS on an Intel/AMD robust architecture) that generates the synthetic flight environment and coordinates the simulation.
- **Sensor Interface Node:** A lightweight edge device (e.g., Raspberry Pi 3, Quad Core 1.2GHz Broadcom BCM2837 64bit CPU) acting as the network gateway for physical position sensors.
- **Test Engineers & System Integrators:** Human operators who configure simulation parameters, inject fault scenarios, and extract telemetry for analysis.

Communication & Edge Devices

- **Middleware (OPCUA / DDS / MQTT):** The data backbone facilitating pub/sub message exchange between the simulator, the SUT, and the sensors.
- **Position Sensors & Electrical Motors:** The physical edge devices providing real-world mechanical feedback and actuation to close the HIL loop.

Networking

- **Distributed HIL Infrastructure:** The network hardware connecting local testbenches with remote simulation nodes, currently relying on standard VLAN and classical VPN isolation.

11.6 Challenges and Needs from PiQASO

To effectively secure this environment, PiQASO must address several domain-specific technical barriers. The expected secure operation process is described in the sequence diagram in Figure 11.4.

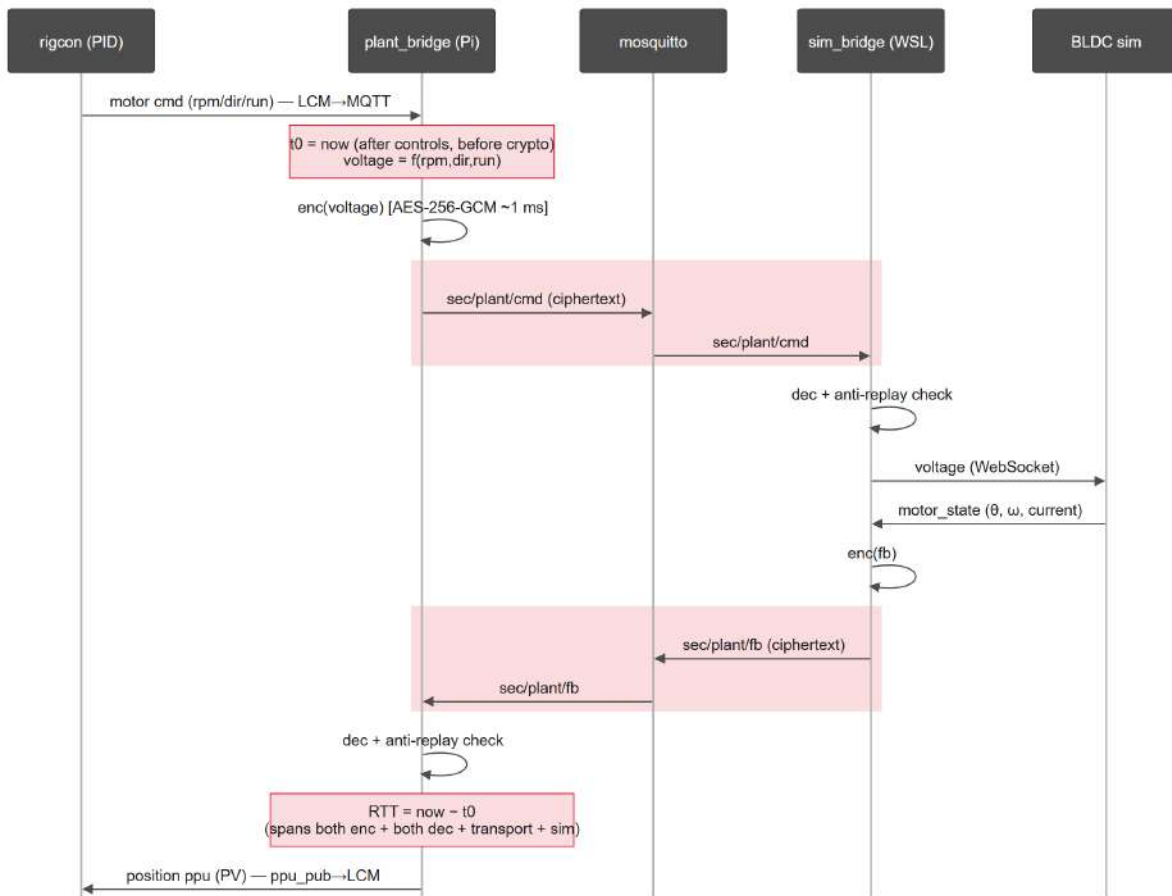


Figure 11.4: HIL operation after PiQASO: one secured control cycle (PQC mode) – command and feedback are sealed end-to-end with the AES-256-GCM session key, and the broker relays ciphertext only.

- **Ultra-Low Latency Overhead:** Cryptographic operations must be exceptionally streamlined. The entire process of signing, encrypting, transmitting, decrypting, and verifying control signals cannot push the total loop time past the 100ms threshold.
- **Transparent Hardware Acceleration:** In aerospace, flight computers undergo rigorous certification (e.g., DO-178C for software, DO-254 for hardware). Modifying the SUT to run PQC natively would invalidate its certification. Therefore, the project must implement FPGA-based PQC acceleration as a “bump-in-the-wire” network proxy that cryptographically secures traffic on behalf of the SUT without altering its native hardware.
- **Secure Component Authentication:** The system requires a scalable, quantum-safe method to verify the identity of newly connected sensors or swapped SUTs before they are permitted to publish to the testbench middleware.
- **Protocol-Agnostic Application:** Security layers must wrap seamlessly around existing industrial protocols (OPCUA/DDS/MQTT) without requiring a ground-up rewrite of the test facility’s established communication architecture.
- **Long-Term IP Secrecy:** Proprietary physics models and flight logs require forward-secret, quantum-resistant sealing to ensure they cannot be decrypted by future adversaries.

11.7 “To-Be” Reference Scenario #1: Quantum-Safe Real-Time Control

The first “To-be” architecture shifts the paradigm from perimeter defense to continuous, application-layer security. By integrating the PiQASO SDK, the testbench anchors trust directly into the simulation payloads. To guarantee that the SUT’s real-time performance is unaffected, cryptographic primitives will be offloaded to an FPGA-based hardware accelerator (QR-TCB) deployed inline.

Purpose, required PiQASO components, and integration point. The purpose of this scenario is to secure the live control loop – the high-frequency exchange of motor commands and sensor feedback – against a quantum-capable adversary while staying inside the aerospace real-time budget. It instantiates the following components of the PiQASO PQC Ensemble: the **PQC SDK** (the `pq_frame` framing layer), which performs an **Authenticated Key Exchange** built on **ML-KEM** (key agreement) and **ML-DSA** (signatures), and protects each message with **AES-256-GCM**; and, optionally, the **QR-TCB** FPGA accelerator to offload the post-quantum primitives. These components integrate at the application layer on the Virtual Testing Computer and the Sensor/SUT node (cf. Section 11.5.1); the broker is untouched and relays ciphertext only.

11.8 “To-Be” Reference Scenario #2: Secure Distributed HIL with IP Protection

Scheduled for the second reporting phase, this scenario builds upon the secured control loop to address multi-node complexity and the protection of data-at-rest. The architecture introduces fine-grained, policy-based access control using Attribute-Based Encryption (ABE). Unlike traditional encryption that targets a specific user, ABE ensures that sensitive data is only legible to personnel who possess specific cryptographic attributes (e.g., Department: Aerodynamics AND Role: Lead), regardless of where the data resides.

Purpose, required PiQASO components, and integration point. The purpose of this scenario is to protect recorded test data at rest and to enforce role-based access to it across distributed engineering teams. It instantiates **Attribute-Based Encryption (ABE)** – used here to realise role-based access control, with roles expressed as cryptographic attributes – supported by **UPKE** for forward-secret archival and the **PQCaaS** Key Management Service for policy and key provisioning. Integration is off the real-time path: the Virtual Testing Computer requests policy keys from the PQCaaS and writes sealed archives to the central repository. This scenario is scheduled for **RP2** and is not in scope for the September milestone.

11.9 “To-Be” Reference Scenario #3: Hardware-Accelerated HIL Control

Purpose, required PiQASO components, and integration point. The purpose of this scenario is to determine how far hardware acceleration can push the secured loop – the upper bound of achievable performance – and to make frequent key rotation affordable. It instantiates the **QR-TCB** FPGA accelerator together with the same **ML-KEM/ML-DSA** key agreement as Reference Scenario #1. The accelerator integrates as a “bump-in-the-wire” inline proxy between the Virtual Testing Computer / SUT and the broker

– the same placement as in Section 11.11.1.1, now executing the post-quantum primitives in hardware rather than software. This scenario belongs to **RP1**.

11.10 Data Model and Transport Pattern

All loop traffic is carried as MQTT topics under the `rtsys/rig/` namespace. Two small JSON messages dominate the real-time loop and are the payloads that PQC protects; their fields are listed in Table 11.2.

Table 11.2: Real-time payloads protected in UC6.1.

Message	Direction	Fields (type) and example
command	control → plant	seq (uint64 counter), voltage (float). e.g. {"seq":353,"voltage":-15.0796}
feedback	plant → control	seq (uint64), theta (angle), omega (angular velocity), current (motor current), all float. e.g. {"seq":353,"theta":1.84,"omega":114.2,"current":0.47}

The data follows three distinct transport patterns, which is what determines where the cryptographic cost falls:

- **Stream:** the `command/feedback` pair is exchanged every control cycle (nominally at 20 Hz) for the duration of a run. The messages are small and latency-critical, so each is sealed individually with the AES-256-GCM session key.
- **Burst:** the ML-KEM/ML-DSA key agreement is a one-shot exchange at session start (and at each re-key). It carries the kilobyte-scale public key, encapsulation ciphertext, and signatures, and sits off the per-cycle critical path – this is where the heavy post-quantum work lives.
- **Chunked (at rest):** recorded test logs are written and sealed at the end of a run (Reference Scenario #2, ABE), not streamed.

Only the `command/feedback` payloads are encrypted on the wire (the secured `sec/*` topics); control toggles and timing telemetry carry no sensor values and remain in plaintext.

11.11 Reference Scenario User Stories and Metrics of Success (KPIs)

11.11.1 User Story #1 (Scenario #1): Secure Control Signal Transmission

UC6.1.US1: Low-Latency Authenticated Motor Actuation Commands to Slat Flap SUT

As a Virtual Testing Computer, I want to send rapid, cryptographically authenticated motor actuation commands to the Slat Flap SUT (System Under Test) so that spoofed commands are prevented, while ensuring commands arrive, are verified, and execute within the strict 100ms aerodynamic timing constraint.

Workflow:

1. **Command Generation:** The Virtual PC generates a dynamic slat extension command based on simulated turbulence.

2. **PQC Processing:** Using the session key established by the ML-DSA-authenticated ML-KEM handshake (Section 11.11.1.1), the PiQASO SDK seals the command payload with AES-256-GCM authenticated encryption. The standardised primitives are **ML-KEM** (FIPS 203) for key agreement and **ML-DSA** (FIPS 204) for signatures.
3. **Transmission:** The secured payload traverses the DDS middleware or MQTT broker.
4. **Hardware Verification (Proxy):** The FPGA inline proxy (acting as a secure gateway for the SUT) intercepts the message, offloading the decryption and signature verification at sub-millisecond speeds.
5. **Execution:** Integrity confirmed, the clean payload is passed over a dedicated short-link to the SUT, which adjusts the physical actuators, logging a completion time well within the 100ms budget.

Expected Benefits

- Completely neutralizes the risk of unauthorized devices injecting false telemetry or malicious motor commands into the testbench.
- Proves the viability of PQC in hard real-time environments through successful hardware acceleration without modifying certified SUTs.
- Maintains the integrity of the aerospace validation process, ensuring that only pristine, untampered data influences the final certification of the flight hardware.

Sequence Diagram. The action workflow above is depicted by the per-cycle data-flow sequence diagram of Section 11.11.1.2 (Figure 11.4); the one-time key agreement that precedes it is shown in Figure 11.5.

Reference Scenarios

1. **Baseline (no PQC):** plaintext control loop – vulnerable to interception, manipulation and replay.
2. **Classical (AES):** symmetric-only payload protection, the pre-quantum baseline.
3. **PQC-enabled:** ML-KEM / ML-DSA key agreement + AES-256-GCM session (the To-Be mode).
4. **Attack scenario:** adversary attempts command spoofing, sensor-feedback tampering, or replay; resilience measured by the block and detection rates in 11.1.

11.11.1.1 Key Agreement: ML-KEM / ML-DSA Handshake

Before any control traffic flows, the two endpoints establish a shared session key through a mutually authenticated, post-quantum handshake. Each side holds a long-term **ML-DSA** signing key pair and trusts the other side's ML-DSA public key (exchanged out of band). The exchange then proceeds as follows:

1. The initiator generates a fresh session **nonce** and a single-use (ephemeral) **ML-KEM** key pair (pk, sk) .
2. Initiator → Responder: $\{pk, nonce\}$, **signed** with the initiator's ML-DSA key.
3. The responder verifies the ML-DSA signature and the freshness of the nonce, then runs **ML-KEM Encapsulate** (pk) , obtaining the encapsulation ciphertext ct and locally deriving the shared secret ss .

4. Responder → Initiator: $\{ct, nonce\}$, **signed** with the responder's ML-DSA key. The value placed on the wire is the encapsulation result ct only; the shared secret ss is never transmitted.
5. The initiator verifies the responder's ML-DSA signature and runs **ML-KEM Decapsulate** (ct, sk) , recovering the same shared secret ss .
6. ss becomes the **AES-256-GCM** session key. Every subsequent command and feedback message is encrypted and authenticated under it, and a monotonic per-message counter rejects replays.

Because the ML-KEM key pair is single-use, the session enjoys **forward secrecy**: a later compromise of a long-term signing key does not expose past sessions. Mutual authentication is provided by the two ML-DSA signatures, and the signed nonce binds the exchange to this session, defeating replay of a recorded handshake. The sequence is shown in Figure 11.5.

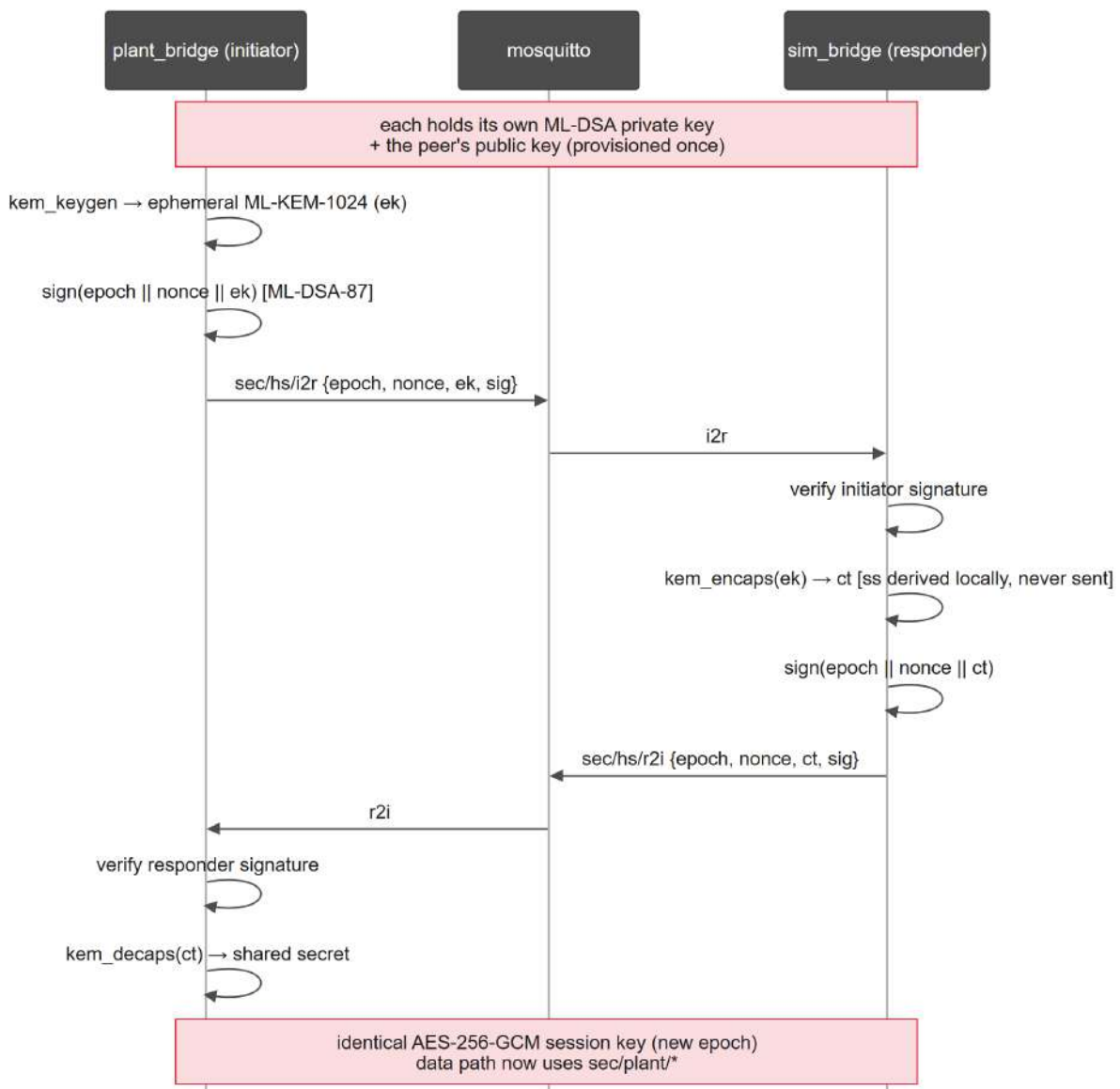


Figure 11.5: Mutually authenticated post-quantum key agreement: an ML-DSA-signed, ephemeral ML-KEM exchange yields the AES-256-GCM session key. The encapsulation ciphertext is the only key material on the wire; the shared secret is derived independently at each end.

On transport security: the project specification calls for **PQ-TLS** as the quantum-resistant transport between distributed nodes; the present demonstrator achieves end-to-end protection at the application layer

instead, framing each payload with `pq_frame` (AES-256-GCM) over ordinary MQTT, which keeps the broker and the existing middleware unmodified.

11.11.1.2 Secured Control Loop – Per-Cycle Data Flow

Once the session key is in place, the control loop runs continuously. In each cycle the controller publishes a command and the plant publishes feedback; with protection enabled, each payload is sealed with the AES-256-GCM session key before it reaches the broker and is opened only at the far endpoint. The broker forwards ciphertext on the `sec/*` topics and never sees a plaintext value. The same loop runs in three selectable cipher modes, which serve as the reference scenarios used for evaluation:

- **none** – plaintext, the functional baseline;
- **AES** – symmetric authenticated encryption with a pre-shared key (classical baseline);
- **PQC** – the post-quantum mode, in which the session key originates from the ML-KEM/ML-DSA handshake of Section 11.11.1.1.

The secured PQC cycle itself is the one shown earlier in Figure 11.4. Because steady-state protection reduces to symmetric AEAD, the per-message overhead is small and roughly constant across the modes; the post-quantum cost is concentrated in the one-time handshake rather than the per-cycle path.

11.11.2 User Story #2 (Scenario #2): Protected IP & Telemetry Logs (ABE = role-based / RBAC)

UC6.1.US2: Quantum-Resistant Archival and Access Control of Classified Simulation Logs

As a System Integrator, I want to archive the results of a highly classified, multi-node simulation run such that the proprietary algorithms embedded in the logs are protected against quantum decryption, and access is restricted accordingly so that standard test technicians cannot extract the core flight logic, while long-term confidentiality of the proprietary algorithms is preserved against future quantum-capable adversaries.

Workflow:

1. **Request:** Upon simulation completion, the Virtual Testing PC requests an archival key from the PiQASO SDK.
2. **Policy Retrieval:** The SDK queries the Key Management Service, which returns a Policy-Based Public Key corresponding to the required access level.
3. **Sealing:** The test logs and recorded algorithmic behaviors are encrypted using the ABE policy (e.g., requiring the attribute “Role: Lead Integrator”).
4. **Storage:** The cryptographically sealed archives are pushed to the central repository.
5. **Selective Decryption:** If a standard technician attempts to access the file, the SDK rejects the request because their attributes do not match the policy. When the System Integrator accesses it, the SDK verifies their cryptographic attributes and successfully reveals the data.

Expected Benefits

- Enables secure, distributed testing across geographic boundaries without exposing the data stream to network-level interception.
- Provides granular, role-based privacy, cleanly separating daily operations from highly sensitive IP management.
- Ensures absolute long-term confidentiality of aerospace research and development assets against the “harvest now, decrypt later” threat.

Sequence Diagram. The archival workflow above is shown in Figure 11.6.

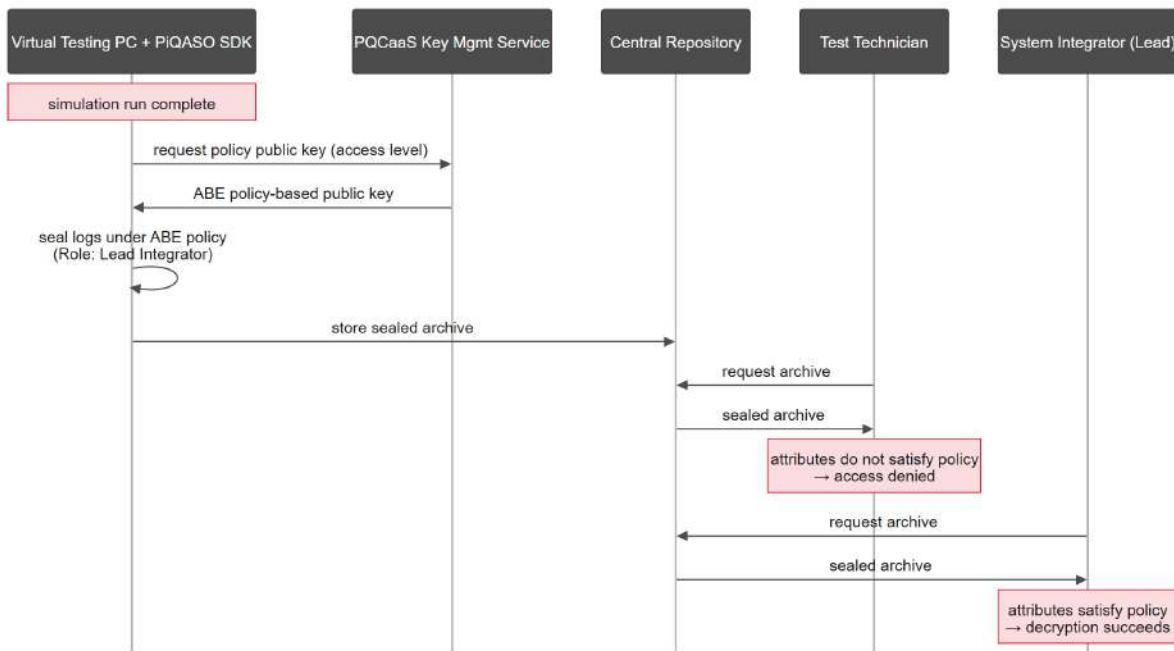


Figure 11.6: ABE archival and selective decryption: test logs are sealed under a role-based attribute policy; only a holder of the matching role attributes can decrypt.

Reference Scenarios

1. **Baseline (no PQC):** logs stored with classical or no encryption – exposed to “harvest now, decrypt later”.
2. **PQC-enabled (ABE / RBAC):** logs sealed under an attribute policy; access granted by role.
3. **Attack scenario:** an unauthorized technician and a quantum-capable adversary attempt access; resilience measured by the access-control and confidentiality KPIs.

11.11.3 User Story #3 (Scenario #3): Bounding HW Acceleration in the HIL Loop

UC6.1.US3: Real-Time Re-Keying and Latency Bound in Secured HIL Loop

As a Test Engineer / System Integrator, I want to establish a fresh session key at every test assignment in the secured Hardware-in-the-Loop (HIL) loop, and determine the upper bound of hardware acceleration achievable within it so that each test run is cryptographically isolated, without the re-keying process breaking the real-time budget.

- **Workflow:** The exchange is identical to the key agreement of Section 11.11.1.1, with two differences: (i) the ML-KEM and ML-DSA operations execute on the FPGA accelerator (running at 270 MHz) rather than in software; and (ii) a full re-handshake (key rotation) is triggered at the start of each test assignment rather than once per session.
- **What is measured:** the minimum key-agreement latency achievable on the accelerator; the maximum loop rate at which per-assignment re-keying still fits the latency budget; and the point at which further reductions in cycle count yield diminishing (exponential rather than linear) returns. Results are reported against the achievable loop rate, since that – not raw cycle count – is what bounds the real-time loop.

Sequence Diagram: (Check Figure 11.5)

Reference Scenarios

1. **Software baseline:** ML-KEM / ML-DSA handshake executed in software.
2. **HW-accelerated:** the same handshake offloaded to the FPGA (270 MHz) – the upper-bound probe.
3. **Per-assignment re-key:** a full re-handshake at each test assignment; report the maximum loop rate that still meets the latency bound.

11.11.4 Application-Specific KPIs

To rigorously validate the success of UC6.1, the following KPIs will be benchmarked during the integration and live simulation trials.

Table 11.3: Application-specific KPIs for UC6.1 (Aerospace Integration).

KPI	Description	Acceptance Criteria
Security		
Command authenticity	Share of spoofed / unauthorized motor commands rejected at signature verification	100% blocked
Sensor-tamper detection	Detection of injected sensor-data tampering	100% detected
Log confidentiality	Resistance of stored logs to quantum key-recovery	No recovery under simulated quantum-capable attack
Real-Time Performance		
Closed-loop response (PQC)	Total Virtual PC → proxy → SUT round trip with full PQC overhead	≤ 100 ms (aerospace hard real-time bound; the per-cycle path is symmetric AEAD)
Key-agreement latency	Time to (re)establish a PQC session (ML-KEM + ML-DSA), one-shot per session / re-key	≤ 150 ms (initial target 100–150 ms)
Per-message crypto	AEAD cost per command / feedback message	$\leq$ a few ms
SUT CPU overhead	Additional CPU load on the certified SUT	0% (external FPGA offload)
Scalability & Stress		
Node scalability	Maximum HIL nodes sustained within the latency budget	≥ 3 distributed nodes (initial target)
Secured throughput	Sustained throughput under PQC payload expansion	No message loss at the nominal loop rate
Stress headroom	Offered load (rate / message size / concurrent streams) at which the loop breaches the bound or drops messages	Report the failure threshold; initial target $\geq 2 \times$ the nominal 20 Hz load
Usability		

Setup time	Time to configure PQC parameters	≤ 5 minutes (cipher mode is a live toggle)
Workflow disruption	Impact on routine classical HIL operations	Zero disruption

Chapter 12

PiQASO UC6.2 - Sector: Aerospace - Quantum-Secure Aviation Maintenance Services Email Communication

12.1 Use Case's Description, Goals and Objectives

The Quantum-Secure Aviation Maintenance Email Communications (QS-AMEC) initiative addresses the critical need for fortified digital correspondence within the high-stakes environment of aviation maintenance and across global Maintenance, Overhaul, and Service (MOS) organizations. In this industry, email serves as the primary artery for the exchange of highly sensitive data, including intricate maintenance schedules, detailed diagnostic reports, critical component replacement records, and real-time operational updates. The sensitivity of this information cannot be overstated, as any compromise in the integrity or confidentiality of these communications could result in severe operational disruptions, financial inefficiencies, or catastrophic safety risks if maintenance crews unknowingly act on manipulated or spoofed directives.

To combat these evolving threats, this use case proposes the integration of Post-Quantum Cryptography (PQC) as a software-defined security overlay directly into existing aviation maintenance email architectures. The implementation of PQC provides a robust defense against both current classical cyberattacks and future threats posed by quantum computing, which threatens to break standard encryption methods. By utilizing localized, client-side quantum-resistant algorithms, the system ensures that long-term maintenance records remain end-to-end encrypted and confidential even against "harvest now, decrypt later" strategies. Similarly, the centralized TAF component acts as an intelligent gatekeeper, evaluating the trustworthiness of incoming messages through a multifaceted lens that includes rigorous cryptographic validation, historical sender analysis, deep metadata inspection, and sophisticated anomaly detection to identify patterns that deviate from established norms.

The current state of the art in aviation maintenance is defined by a heavy reliance on S/MIME and OpenPGP protocols to secure the "digital thread" of airworthiness data¹. In daily operations, most Maintenance, Overhaul, and Service (MOS) organizations utilize these classical cryptographic standards to exchange sensitive Technical Orders, Job Cards, and release-to-service documentation. However, this established landscape is increasingly supplemented by AI-driven behavioral analytics at the mail gateway level to detect sophisticated phishing and impersonation attempts that bypass traditional signature

¹IETF draft-ietf-openpgp-pqc: The technical roadmap for updating the OpenPGP standard to support quantum-resistant key encapsulation.

verification ².

As the industry confronts the “harvest now, decrypt later” threat, a critical transition is underway toward Hybrid Post-Quantum Cryptography (PQC) ³. This approach integrates NIST-standardized algorithms like ML-KEM (Kyber) alongside existing transport infrastructure, ensuring that transport channels remain quantum-safe and long-term maintenance records remain confidential even as quantum computing capabilities evolve ⁴. Furthermore, the adoption of Identity-Based Filtering (IBF) and BODIS (Behavioral Outlier Detection in Signals) is emerging as a benchmark for verifying sender intent, effectively bridging the gap between digital encryption and physical operational safety ⁵.

The core objectives of the QS-AMEC framework are centered on creating a zero-trust software-defined environment for data exchange in aviation. By securing communication channels with hybrid quantum-resistant transport protocols and enforcing end-to-end payload encryption at the client level, the framework guarantees that only authorized entities can access or alter sensitive maintenance logs. Furthermore, by evaluating the reliability of messages before they are processed, the system effectively filters out malicious content, such as impersonation attempts, unauthorized data injections, or subtle message manipulations that could lead to improper aircraft servicing. This comprehensive approach ensures that the exchange of information between internal employees and external MOS partners is both secure and reliable, maintaining a seamless digital thread of airworthiness documentation.

Ultimately, the deployment of this integrated solution empowers maintenance managers and technicians to make timely informed decisions with the confidence that the data they are viewing are authentic and untampered with. By drastically reducing exposure to malicious or compromised messages, QS-AMEC minimizes the risk of human error derived from fraudulent instructions. This increased level of security not only protects the intellectual property and operational continuity of the organizations involved, but also upholds the overarching goal of passenger safety by ensuring that every repair and inspection is guided by verified, high-integrity information.

12.2 “As-Is” Scenario

A maintenance engineer, using a standard email client, composes a message containing maintenance reports or component requests. The email client connects to the company’s email server using standard protocols such as SMTP for sending and IMAP/POP3 for receiving. The server then delivers the message to a single recipient within the company or to multiple recipients in other MOS companies. The email client relies on the server to manage message storage, delivery, and retrieval, but no verification is performed on the content or sender beyond basic authentication and standard transport encryption. Crucially, the message payload remains in plaintext while at rest on the email server, exposing sensitive maintenance data to potential internal breaches.

The email client allows 1-to-1 and 1-to-many communication. In 1-to-1 exchanges, a single recipient receives the message directly from the company’s email server. In 1-to-many exchanges, the server handles multiple recipients, often via distribution lists or manual inclusion of multiple addresses. Regardless of mode, messages are typically protected only in transit using standard transport encryption (TLS), and sender authentication is based on username/password or legacy PKI certificates. Because standard

²EASA Part-IS & ED-203A: Regulatory frameworks mandating maintenance organizations move toward comprehensive aeronautical information security.

³NIST FIPS 203 & 204: Finalized standards for ML-KEM and ML-DSA, currently being integrated into aviation-grade secure email gateways.

⁴FAA PQC Roadmap: Institutional guidance driving the transition of National Airspace (NAS) business systems toward a quantum-secure architecture.

⁵ICAO Doc 10188: Global strategy for protecting the civil aviation ecosystem against large-scale computational threats to data integrity.

TLS relies on classical cryptography, all intercepted data in this scenario are currently vulnerable to “Harvest Now, Decrypt Later” quantum attacks. Furthermore, no advanced verification of message integrity beyond basic cryptographic signatures is performed.

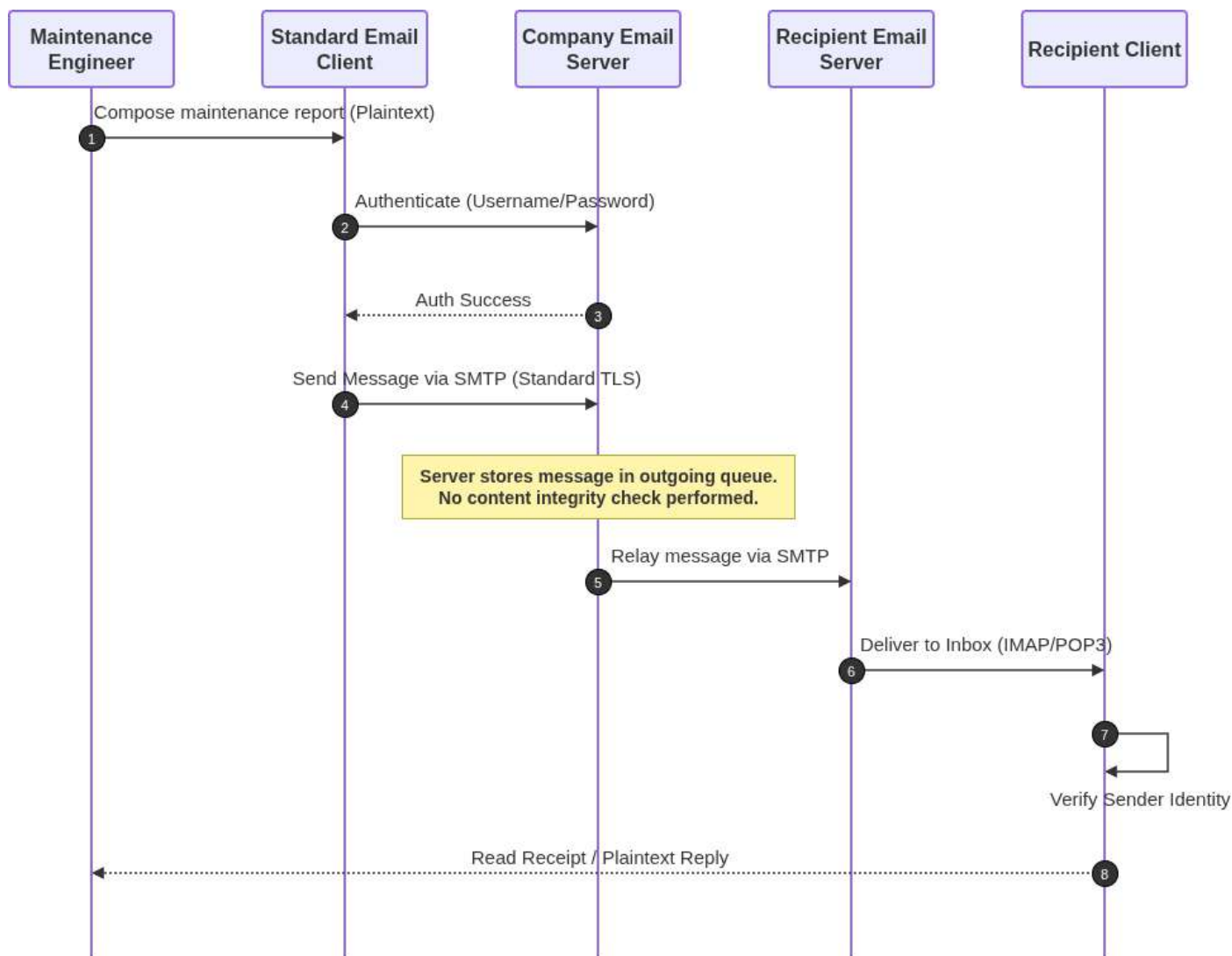


Figure 12.1: Functional description of email client-server communication in the “As-Is” scenario.

This scenario highlights several critical limitations of the current system. Although authentication ensures that the sender is a valid account holder, it does not provide guarantees regarding the correctness, completeness, or intent of the message content. An event of misbehaviour could occur, for example, if an email is manipulated, spoofed, or erroneously sent to multiple recipients, potentially leading to operational inefficiencies or misinformed maintenance actions. Because there is no End-to-End Encryption (E2EE), intermediate nodes can read or alter the payload.

The efficacy of communication depends entirely on the ability of recipients to manually verify the content and authenticity of the incoming messages. Without automated verification, maintenance personnel rely on procedural checks or prior knowledge of the sender, leaving the system highly vulnerable to phishing, impersonation, and message tampering.

In summary, current email communication in aviation maintenance involves standard email clients and servers that support 1-to-1 and 1-to-many communication. Although it provides basic authentication and transport encryption, it lacks dynamic verification, quantum-resistant payload protection, and protection against manipulated or malicious content. As-Is serves as the baseline for implementing the PiQASO Server and a Client-Side Cryptographic Module to secure maintenance communications.

12.3 Components and roles considered for UC6.2

In the following text, we describe the main components and roles considered in this use case. The fundamental enabler of secure email communication in aviation maintenance is the combination of standard email interfaces paired with a Client-Side Cryptographic Module, software-defined email servers, and Post-Quantum Cryptography (PQC), which ensures end-to-end confidentiality, integrity, and authenticity.

To provide a clear visual roadmap for this ecosystem, **Figure 12.2** illustrates the architecture of the high-level system. Highlights the strict boundary between the PQC-secured internal network (Layer 1) and the classical legacy infrastructure of external partners (Layer 2), showing how these components interact to support both internal 1-to-1 and external 1-to-many workflows.

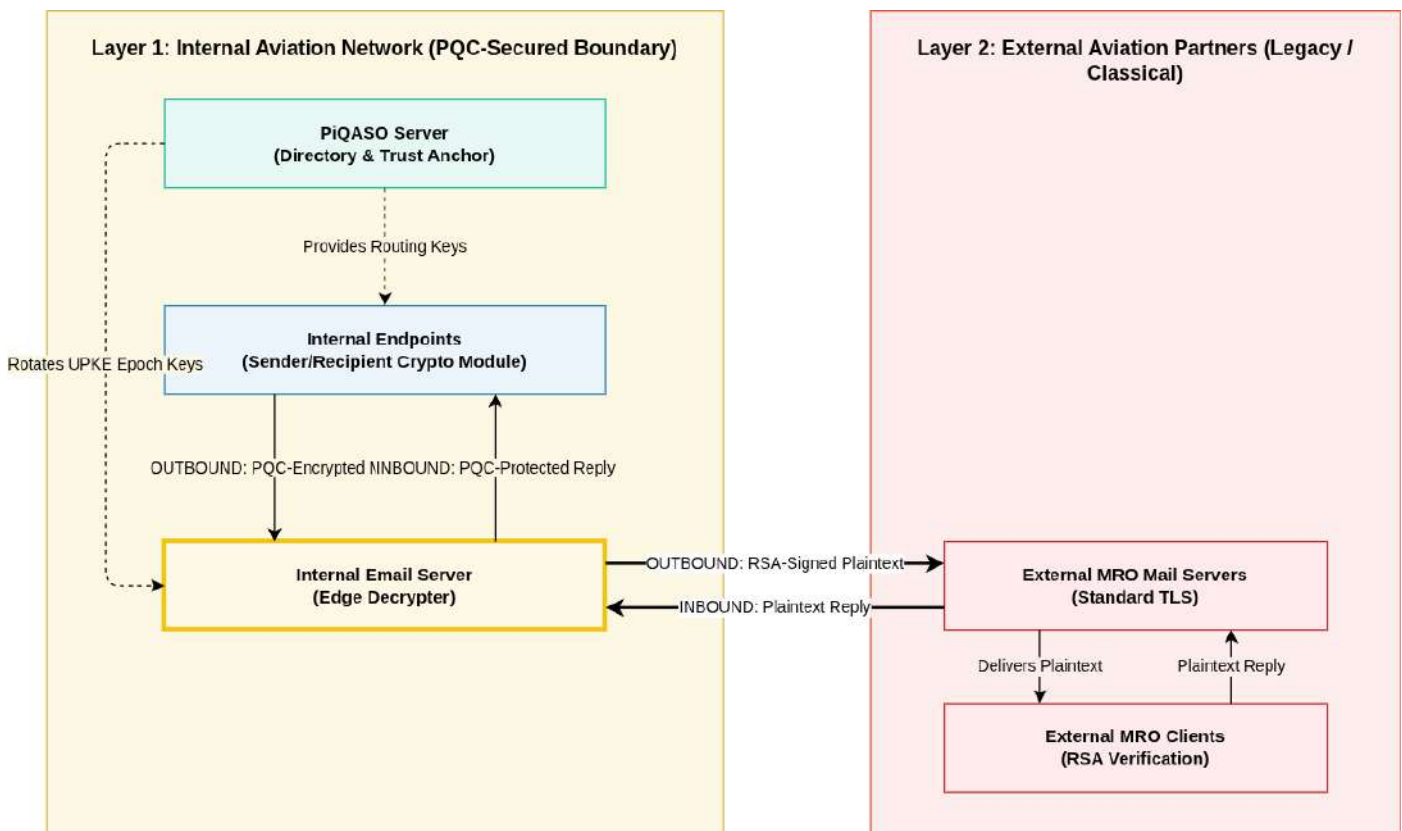


Figure 12.2: High-Level Architecture of the QS-AMEC Integration, illustrating internal PQC boundaries and the Internal Email Server acting as an Edge Decrypter.

The specific components supporting these workflows are detailed below:

- **Sender and Recipient Email Clients & Cryptographic Modules:** Standard email clients (or webmail interfaces) are used by employees or MOS company personnel to compose, send, receive, and read emails. They are paired with a Client-Side Cryptographic Module (deployed as a local proxy or secure webmail engine) that prepares messages for PQC encryption. This module generates random session keys for each message, encapsulates them using the appropriate PQC public keys, and decrypts incoming messages, ensuring that all cryptographic operations occur in secure software memory.
- **Internal Email Server (Dual-Role MTA & Edge Decrypter):** The company's central email server manages message transport and routing policies, operating in two distinct modes. For internal communications, it acts as a blind relay, transparently routing end-to-end PQC-encrypted payloads to internal mailboxes. However, for external communications, it operates as an **Edge Decrypter**. It handles its

own Updatable Public Key Encryption (UPKE) epoch keys, natively decrypts outbound PQC payloads, applies classical RSA Signatures, and bridges the gap to the legacy internet.

- **The PiQASO Server (Directory and Trust Anchor):** This centralized node acts as the core cryptographic authority for the architecture. It serves as the PQC Key Directory (PQCaaS), managing public/private key provisioning and authentication. The sender cryptographic modules retrieve the verified ephemeral PQC public keys of internal recipients—or the UPKE epoch keys of the Internal Email Server—directly from this service to ensure secure message routing.
- **External Recipient Servers & Clients:** External or partner recipient servers receive legacy-translated plaintext messages from the Internal Email Server via standard classical TLS. They rely purely on classical certificate verification (RSA Signatures) and are responsible for normal message storage and delivery to the final external clients. They do not require any PQC infrastructure.
- **Message Flow Summary:**
 - **Internal 1-to-1 communication:** The sender's cryptographic module encrypts the email payload with a session key, encapsulates it with the recipient's PQC public key (fetched from the PiQASO Server), and sends it via the Internal Email Server. The server blindly relays the package, and the recipient's module decapsulates the key and decrypts the message locally.
 - **External 1-to-many communication (Legacy Translation):** For recipients outside the secure PQC infrastructure, the sender's module specifically encrypts the message using the Internal Email Server's UPKE epoch key. The Internal Email Server natively decapsulates the key and decrypts the payload in its secure memory, destroying the symmetric key to ensure Forward Secrecy. It then applies a classical RSA Signature, removes the PQC layer, and relays the plaintext message via standard TLS to multiple external recipient servers. External responses follow a reverse protected workflow, where the Email Server intercepts the plaintext reply and PQC-encrypts it before routing it internally.

12.4 Challenges and Needs from PiQASO

These scenarios collectively address the critical challenge of ensuring that aviation maintenance personnel can reliably act on email communications, even in environments where sender trust or message integrity cannot be assumed. By combining PQC encryption with dynamic verification, QS-AMEC ensures that operationally critical messages are both secure and trustworthy, enabling informed decisions without undue risk.

The efficacy of aviation maintenance operations is highly dependent on the correctness and trustworthiness of internal and inter-company email communications. Emails contain critical information, such as maintenance schedules, diagnostic reports, and operational instructions. However, in the current "As-Is" scenario, email messages traverse standard email clients and servers without advanced verification mechanisms. Sender authentication and TLS encryption provide only basic protection, leaving the system vulnerable to impersonation, message tampering, or misrouting, which could compromise operational efficiency or safety. Consequently, it becomes essential to introduce mechanisms to evaluate the trustworthiness of incoming messages before acting on their content. This requires a framework capable of dynamically combining cryptographic verification (ensuring message integrity and authenticity) with contextual analysis.

In the current scenario, a sender composes an email using an email client, which relies on the SMTP server to deliver the message to one or multiple recipients. Each recipient receives the message in plaintext after standard TLS encryption is applied in transit. Although transport encryption protects against

intercept, it does not provide end-to-end security or ensure the integrity of message content once it reaches the recipient. As a result, recipients have no automated mechanism to evaluate the reliability or correctness of the message, relying solely on procedural checks or prior trust in the sender.

To address this, the integration of Post-Quantum Cryptography (PQC) and the PiQASO Server introduces a comprehensive software-based security overlay. The client-side cryptographic module prepares the message for end-to-end PQC encryption, generates a symmetric session key, and encapsulates it with the public PQC key of the recipient, ensuring that only the intended recipient can decrypt and verify the message. This approach addresses two primary needs: real-time verification and payload protection. The TAF allows the recipient client (or the centralized gateway) to evaluate the cryptographic trustworthiness of each incoming message in real-time. This is particularly important in 1-to-many communications, where each recipient may receive a slightly different message version encapsulated for their own ephemeral key, and each must be independently verified.

To evaluate these inputs, the TAF integrates multiple pieces of cryptographic evidence. For example, trust depends on successful PQC decryption, the mathematical validity of post-quantum digital signatures, and consistency across similar messages sent to multiple recipients. This enables the system to flag unusual or mathematically inconsistent messages without discarding them outright, preserving operational information while signaling limited trust when necessary.

In practice, two reference workflows dictate the challenges for QS-AMEC: minimal-intrusion integration for internal 1-to-1 communication and legacy translation for external 1-to-many scenarios. In the internal scenario, the verification is anchored in the Client-Side Cryptographic Modules (software proxies or secure webmail engines). Both the sender and recipient modules authenticate with the central PQCaaS to retrieve keys. The message is encrypted end-to-end and routed via standard internal transport. Trust is verified locally when the recipient's module uses its own PQC libraries to ensure the content is authentic. In the external scenario, the sender encrypts the message using the PiQASO Gateway's key. The internal SMTP server routes this over a secure PQ-TLS tunnel to the Gateway, which decrypts it, applies classical RSA Signatures, and sends the message over standard TLS to outside recipients to ensure security within the company perimeter while maintaining legacy interoperability.

The primary challenge is the seamless transition from classical to quantum-resistant infrastructure without disrupting existing maintenance workflows. This requires the framework to handle hybrid cryptographic states strictly through software deployment. To validate these capabilities, the QS-AMEC use case will be deployed across simulated MRO environments, featuring typical BYOD (Bring Your Own Device) network constraints and cloud-hosted data exchanges to stress-test the real-time processing limits of the software-based. The high-level requirements include the establishment of a software-based Quantum-Resistant Root of Trust (PQCaaS), the ability to perform low-latency cryptographic operations at the edge, and the maintenance of a tamper-proof audit log for all verification decisions.

To meet these requirements, selected PiQASO functionalities are mapped across specific reporting periods to ensure a phased, software-defined rollout within the aviation maintenance infrastructure.

Table 12.1: PiQASO Functionality Rollout Roadmap for QS-AMEC

PiQASO Functionality	Description and Purpose in the Use Case	RP1	RP2
PQ Signatures	Enables post-quantum digital signatures (ML-DSA) to authenticate senders and guarantee message integrity without relying on classical PKI.	✓	
PQ KEM	Facilitates end-to-end encryption by securely encapsulating symmetric session keys using lattice-based algorithms (ML-KEM) at the client level.	✓	

PQ-TLS	Secures the internal transport tunnel between the standard SMTP server and the PiQASO translation gateway, protecting data during the legacy routing loop.	✓	✓
--------	--	---	---

By aligning these strictly software-defined functionalities with the maintenance lifecycle, the QS-AMEC framework ensures that as the underlying algorithms mature, the aviation sector remains protected against both classical spoofing and future quantum threats. This roadmap allows PiQASO to provide a scalable solution that transforms email from a vulnerable communication tool into a verified, high-integrity asset.

12.5 "To-Be" Reference Scenario #1: Minimal-Intrusion Integration (Internal Communication)

Description: This scenario establishes a high-security, software-defined workflow between authenticated internal endpoints within a single aviation organization, with a primary focus on securing **1-to-1 email exchanges**. By integrating quantum-based cryptographic operations at the edge, the system ensures quantum-resistant end-to-end security while maintaining the existing internal email server as a transparent relay. To satisfy diverse operational and compliance requirements, the architecture is crypto-agile, supporting two distinct key-establishment profiles: an **Edge-Driven mode** (via the local SDK) for strict Zero-Trust, and a **Service-Driven mode** (via the PQCaaS) for centrally managed key brokerage. Furthermore, this architecture specifically counters "Harvest Now, Decrypt Later" (HNDL) threats by guaranteeing **Forward Secrecy** through the generation and subsequent destruction of a unique ephemeral symmetric session key for every individual message.

Workflow:

- Prerequisite (UPKE Epoch Keys):** To ensure Forward Secrecy, endpoints continuously rotate their PQC Epoch Keys within the central PiQASO directory to make sure that they are available for peer lookup.
- Authentication:** The sender's cryptographic module authenticates with the centralized PiQASO Server using PQC-compliant digital signatures (e.g., ML-DSA) to establish a trusted session before requesting key materials.
- Composition:** The sender composes the email in plaintext within their standard email client interface, which remains isolated from cryptographic operations.
- Agile Key Establishment:** Depending on the active security policy, the system executes one of two paths for the 1-to-1 exchange:
 - Mode A (Local SDK/Edge-Driven):* The sender's module requests the recipient's current PQC Epoch Key from the directory, generates a random symmetric session key (K_{msg}) locally, and encapsulates it via ML-KEM (C_{key}).
 - Mode B (Centralized KMS/Service-Driven):* The sender's module requests a session key directly from the PiQASO Server. The server acts as an active Key Management Service (KMS), generating K_{msg} and delivering it to both parties through secure PQ-TLS tunnels.
- PQC-Based Encryption:** Local software encrypts the message content (C_{msg}) using the established symmetric key (K_{msg}) via AES-GCM. Performing these operations within secure software memory ensures absolute isolation from the legacy email application.

6. **Secure Transmission:** The client submits the protected payload (along with the encapsulated key or a KMS Session ID) to the internal Email Server via SMTP over a standard TLS tunnel.
7. **Agile Decryption:** Upon receipt, the recipient's module retrieves K_{msg} by locally decapsulating C_{key} with their private epoch key (Mode A) or by requesting the specific key from the PiQASO Server via PQ-TLS (Mode B). The module then decrypts C_{msg} in plaintext.
8. **Forward Secrecy Execution:** Immediately after successful decryption and local storage handoff, the ephemeral session key (K_{msg}) is permanently destroyed from local device memory (RAM).
9. **Secure Display:** The recipient views the authentic and PQC-secure plaintext email while the module verifies all cryptographic signatures.

Objectives:

- Ensures **Forward Secrecy** by using a unique, ephemeral symmetric session key for each 1-to-1 email exchange, ensuring that future compromises of endpoint devices do not expose historical network traffic to HNDL adversaries.
- Demonstrate architectural agility by supporting both decentralized (SDK epoch key encapsulation) and centralized (PQCaaS PQ-TLS delivery) key distribution models.
- Implement "Key-over-Session-Key" protection without requiring legacy internal server-side upgrades.
- Ensure mandatory endpoint authentication through a centralized PQC trust provider, eliminating the risk of internal impersonation or lateral movement.
- Minimize latency by using optimized software libraries for lattice-based computations, ensuring maintenance workflows are not disrupted.

Components & Roles:

- **Sender/Receiver Email Client & Cryptographic Module:** A dual-component endpoint where the standard Email Client acts purely as the UI. It is paired with the local Cryptographic Module (SDK), which dynamically handles both local ML-KEM encapsulation (Mode A) and PQ-TLS KMS communications (Mode B).
- **PiQASO Server (PQCaaS):** Acts as the organizational trust anchor. Depending on the operational profile, it serves as either a secure public key directory managing UPKE rotations (Mode A) or an active Key Management Service (KMS) generating and distributing session keys (Mode B).
- **Internal Email Server:** Operates as a legacy-compatible relay for standard transport without ever needing access to the unencrypted message content or symmetric keys.

12.6 “To-Be” Reference Scenario #2: Extended-Message Integration (External Communication)

Description:

This scenario defines the architecture for extended integration between an internal PQC-enabled network and external aviation partners, with a specific focus on **1-to-many email broadcasts**. The primary

motivation for this architecture is the legacy constraint that outside mail servers do not support Post-Quantum Cryptography (PQC). To bridge this gap, the architecture upgrades the Internal Email Server to act as an "Edge Decrypter" assisted by the PiQASO Server acting as a cryptographic coprocessor. When an internal user sends a broadcast to multiple external partners, the PiQASO Server handles the heavy lifting of generating and encapsulating the session keys specifically targeted at the Email Server's Epoch Key. The Email Server relies on PiQASO for decapsulation, performs native server-side decryption, evaluates the payload, and re-signs the content with a classical RSA Signature before routing it externally. To protect the internal leg of the network against "Harvest Now, Decrypt Later" (HNDL) threats, the Email Server utilizes Updatable Public Key Encryption (UPKE), continuously rotating its keys on a per-epoch basis to ensure Forward Secrecy.

Workflow:

1. **UPKE Prerequisite:** The Internal Email Server continuously rotates its PQC Epoch Keys within the PiQASO directory to ensure Forward Secrecy for internal-to-server transit.
2. **Composition & Request:** Sender composes a 1-to-many broadcast email intended for external recipients. The local Crypto Module requests routing keys from the PiQASO Server, sending the recipient IDs.
3. **Legacy Identification & Coprocessor Encapsulation:** PiQASO Server identifies the target domains as "Legacy" (missing PQC support). Acting as a coprocessor, it generates a symmetric session key (K_{msg}), encapsulates it with the current PQC Epoch of the Internal Email Server to create C_{key} , and returns both to the Sender.
4. **Targeted Local Encryption & Submission:** The Sender's module encrypts the payload (AES-GCM) with K_{msg} to create C_{msg} , destroys K_{msg} locally, and submits the package $\{C_{msg}, C_{key}\}$ to the Email Server via standard TLS.
5. **Decapsulation Request:** The Internal Email Server receives the package and requests the PiQASO Server to decapsulate C_{key} .
6. **Coprocessor Decapsulation:** The PiQASO Server decapsulates C_{key} using the appropriate epoch key and returns K_{msg} to the Email Server.
7. **Server-Side Decryption & Translation:** Acting as the Edge Decrypter, the Email Server decrypts the payload to plaintext, permanently destroys K_{msg} from RAM (Forward Secrecy) and applies a Classical RSA Signature.
8. **External Relay:** The Email Server relays the RSA-signed plaintext via standard external TLS to the external legacy servers, which deliver the emails to the recipients' inboxes for standard verification.
9. **Protected Response Loop:** External replies are received in plaintext via standard TLS. The Internal Email Server requests PQC-protection from the PiQASO Server, which generates a new session key and encapsulates it with the internal recipient's PQC key. The Email Server then encrypts the incoming reply and delivers a PQC-protected response back to the internal user.

Objectives:

- Address the primary legacy constraint by facilitating secure 1-to-many communication with external MOS organizations whose mail servers lack PQC infrastructure.
- Implement UPKE (per-epoch keys) directly at the internal mail routing layer to ensure the internal network segment is protected by Forward Secrecy.

- Streamline computational overhead by offloading the KEM generation, encapsulation, and decapsulation processes to the PiQASO Server acting as a cryptographic coprocessor.
- Ensure a secure, PQC-protected internal return path for incoming external responses.

Components & Roles:

- **Sender Crypto Module:** Dynamically adapts to multiple external recipients by encrypting outbound legacy-bound broadcasts using keys actively generated by the PiQASO Server.
- **Internal Email Server (Edge Decrypter):** Upgraded to manage UPKE epoch keys. It performs server-side payload decryption, RSA signing for external delivery, classical routing, and payload encryption for incoming replies.
- **PiQASO Server (Directory & Coprocessor):** Acts as the organizational trust anchor and cryptographic coprocessor. Identifies legacy domains, generates session keys (K_{msg}), and handles all KEM encapsulation and decapsulation operations on behalf of the clients and the Email Server.
- **External Legacy Servers & Clients:** Manage standard external message reception, relying purely on classical certificate verification (RSA Signatures).

12.7 Reference Scenario User Stories and Metrics of Success

12.7.1 User Story #1 (Reference Scenario #1): Secure Internal Email Exchange

(NEDHO Collaborator – Colleague)

UC6.2.US1a: Confidential and Authentic Internal Scheduling Updates

As a NEDHO Collaborator, I want to send internal scheduling updates to another collaborator in a way that ensures confidentiality and authenticity so that scheduling information remains protected from unauthorised disclosure and is verifiably trustworthy as originating from a legitimate sender.

UC6.2.US1b: Quantum-Resistant, Forward-Secure Message Reception

As a NEDHO Collaborator, I want to receive messages from other collaborators that are verifiable as authentic, secure against future quantum threats, and protected by Forward Secrecy so that message authenticity is guaranteed and past communications remain protected even if long-term keys are compromised in the future.

User Story Confirmation

The system facilitates a crypto-agile, 1-to-1 email exchange secured with **Post-Quantum Cryptography (PQC)**. Based on operational policies, the workflow supports two distinct key-establishment modes:

- **Mode A (Edge-Driven / Local SDK):** The sender's local Cryptographic Module generates a PQC-secure session key, encapsulates it using the recipient's continuously rotated Epoch Key, and encrypts the message locally. The recipient decapsulates it locally.
- **Mode B (Service-Driven / Central KMS):** The PiQASO Server acts as an active Key Management Service, generating and securely delivering the symmetric session key to both the sender and recipient over a PQ-TLS tunnel.

- **Network level:** In both modes, the encrypted message payload is transmitted via the internal Email Server over standard TLS, ensuring that the email server remains completely blind to the contents.
- **Forward Secrecy:** In both modes, the symmetric session key is permanently destroyed from local memory immediately after encryption/decryption.

User Story Workflows

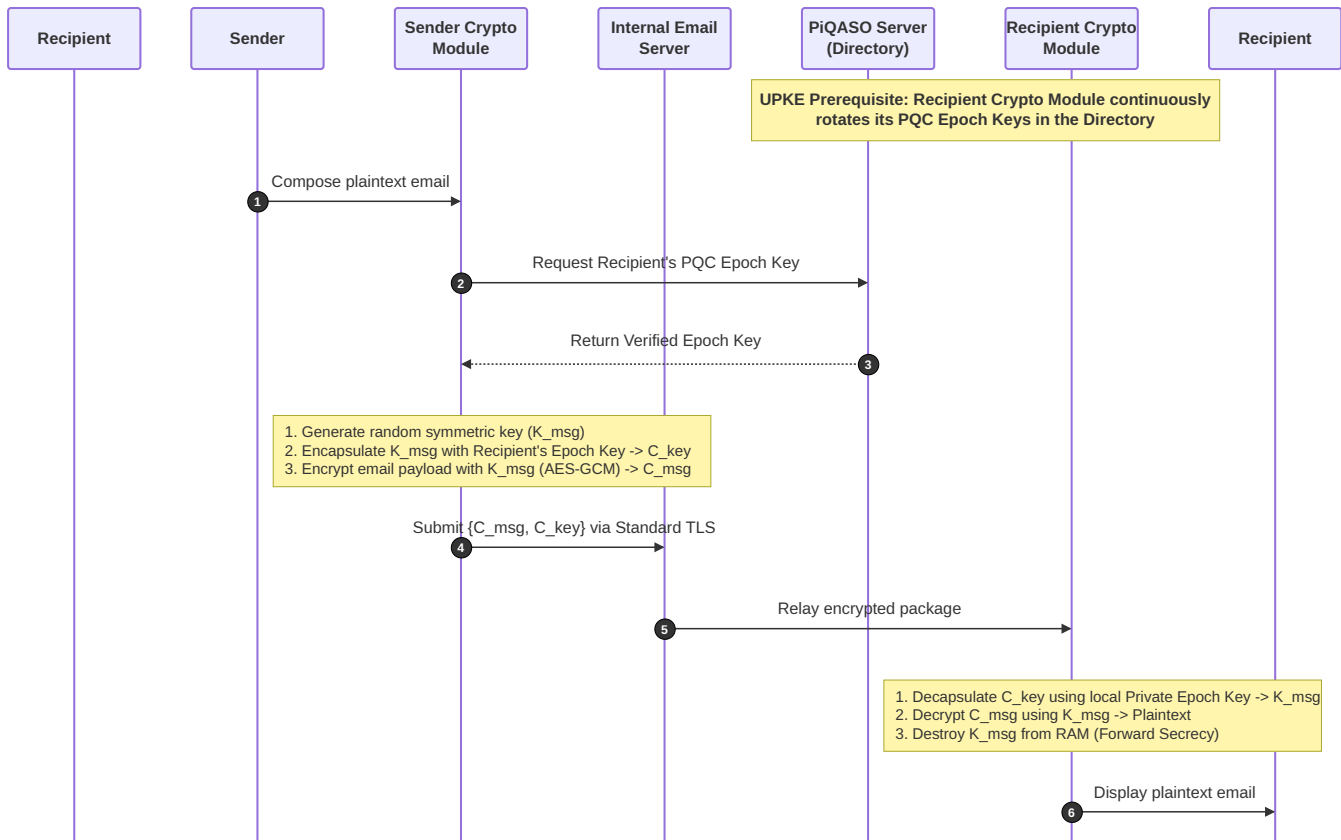


Figure 12.3: Sequence diagram for PQC-secured email exchange - Mode A (Local SDK)

Mode A (Local SDK) Sequence Description:

1. **UPKE Prerequisite (Async):** The Recipient's Crypto Module continuously rotates its PQC Epoch Keys within the PiQASO Server directory.
2. **Email Composition:** The Sender composes a plaintext email.
3. **Key Retrieval:** The Sender Crypto Module requests and receives the Recipient's verified PQC Epoch Key from the PiQASO directory.
4. **Edge Encryption:** The Sender's module generates a random symmetric key (K_{msg}), encapsulates it with the Recipient's Epoch Key to create C_{key} , and encrypts the email payload (AES-GCM) to create C_{msg} .
5. **Transmission & Relay:** The Sender submits $\{C_{msg}, C_{key}\}$ to the Internal Email Server via standard TLS, which relays it to the Recipient.
6. **Edge Decryption:** The Recipient Crypto Module decapsulates C_{key} using its local Private Epoch Key to retrieve K_{msg} , decrypts C_{msg} to plaintext, and permanently destroys K_{msg} from RAM to ensure Forward Secrecy.

7. **Display:** The Recipient views the plaintext email.

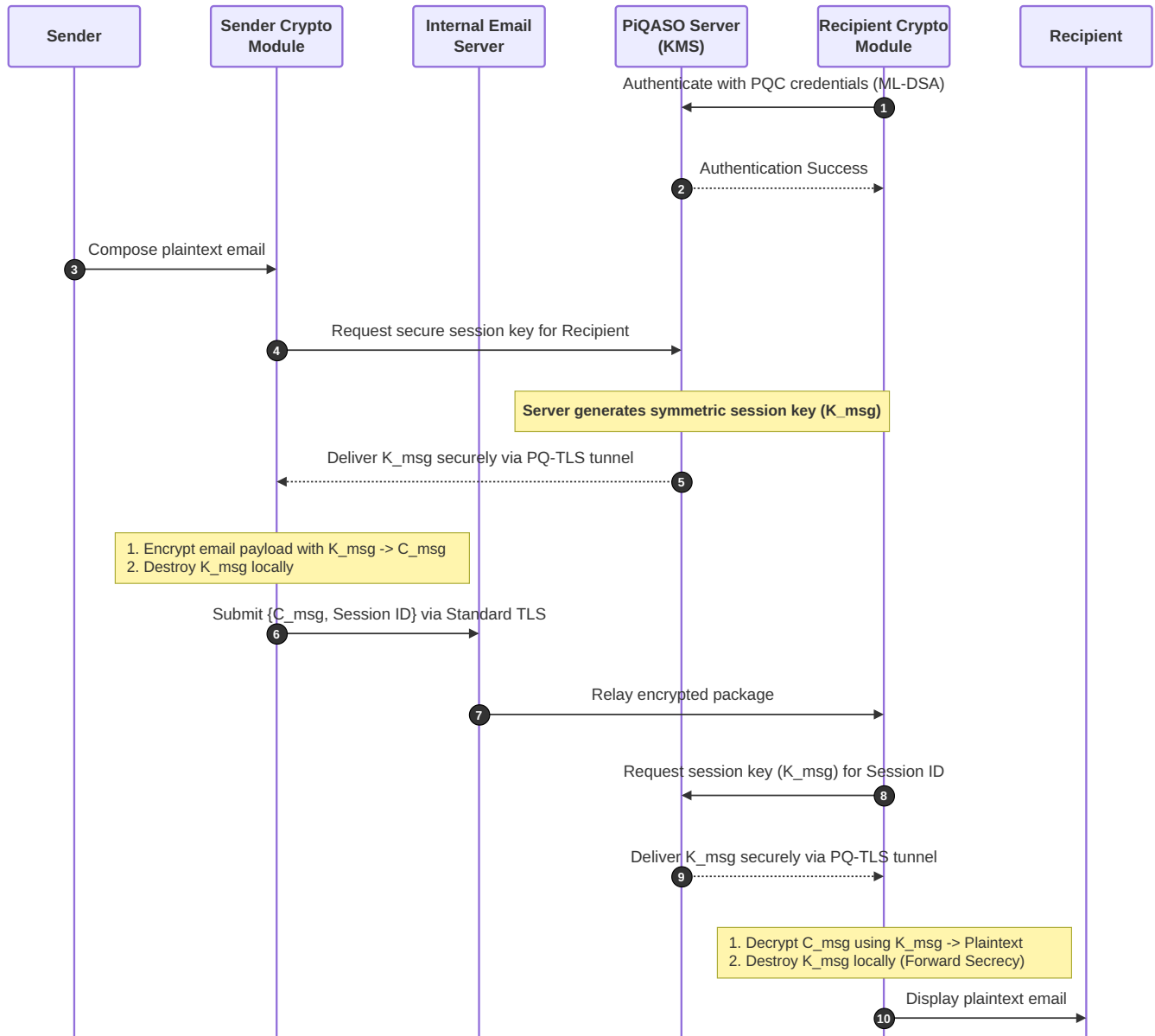


Figure 12.4: Sequence diagram for PQC-secured email exchange - Mode B (Central KMS)

Mode B (Central KMS) Sequence Description:

1. **Authentication:** The Sender Crypto Module authenticates with the PiQASO Server using PQC credentials (ML-DSA) and successfully establishes a trusted session.
2. **Email Composition:** The Sender composes a plaintext email within their local client.
3. **Key Request (Sender):** The Sender Crypto Module requests a secure session key for the specific recipient from the PiQASO Server.
4. **Key Delivery (Sender):** The PiQASO Server generates a symmetric session key (K_{msg}) and delivers it to the Sender via a secure PQ-TLS tunnel. (*Local Action: The Sender encrypts the payload to create C_{msg} and destroys K_{msg} locally.*)

5. **Secure Submission:** The Sender submits the encrypted payload (C_{msg}) alongside a unique Session ID to the Internal Email Server via standard TLS.
6. **Internal Relay:** The Email Server relays the encrypted package to the Recipient.
7. **Key Request (Recipient):** The Recipient Crypto Module uses the received Session ID to request the corresponding K_{msg} from the PiQASO Server.
8. **Key Delivery (Recipient):** The PiQASO Server securely delivers K_{msg} to the Recipient via a PQ-TLS tunnel. (*Local Action: The Recipient decrypts C_{msg} to plaintext and permanently destroys K_{msg} locally to guarantee Forward Secrecy*).
9. **Display:** The Recipient views the plaintext email.

Performance depends on:

- Integration and compliance of the cryptographic module with the ML-KEM / ML-DSA standards.
- The latency overhead introduced by local decapsulation (Mode A) versus PQ-TLS network requests (Mode B).
- Resilience against attack vectors (man-in-the-middle, forged sender identity, quantum "Harvest Now, Decrypt Later" attacks).

Reference Scenarios for Evaluation:

1. **Baseline (no PQC):** Standard TLS-secured email (vulnerable to future quantum attacks).
2. **Mode A Evaluation:** Sender-receiver exchange securely managed via the SDK with local UPKE epoch key encapsulation.
3. **Mode B Evaluation:** Sender-receiver exchange securely managed via PQCaaS centralized key generation.

12.7.2 User Story #2 (Reference Scenario #1): Secure External Email Exchange

(NEDHO Representative – Collaborating MRO Companies)

UC6.2.US2a: Secure Multi-Party Transmission of Operational and Maintenance Information

As a NEDHO Representative, I want to securely transmit operational and maintenance information to multiple collaborating MRO Companies at the same time, ensuring confidentiality, authenticity, and integrity of internal communication so that sensitive operational data reaches all intended recipients without being disclosed, altered, or forged by unauthorised parties.

UC6.2.US2b: Authentic, Legacy-Compatible Reception of NEDHO Communication with Forward Secrecy

As a Collaborating MRO Company, I want to receive authentic NEDHO communication that is compatible with legacy protocols so that communication authenticity is verified while maintaining interoperability with existing systems, and the internal NEDHO network remains protected from unauthorized access through Forward Secrecy.

User Story Confirmation

The system enables **secure 1-to-many communication** by bridging internal Post-Quantum Cryptography (PQC) with external legacy standards. The Internal Email Server serves as a specialized Edge Decrypter, while the PiQASO Server acts as a cryptographic coprocessor.

We can distinguish the following phases:

- **Internal Sender side:** The sender composes an email broadcast. The local Crypto Module requests routing keys from the PiQASO directory. PiQASO identifies the legacy targets, generates a session key, encapsulates it with the UPKE Epoch Key of the Email Server, and returns the key materials. The Sender encrypts the payload and submits it.
- **Network & Translation level:** The Email Server receives the package and de-capsulates the session key to the PiQASO Server. It then performs native server-side decryption, destroys the symmetric key to ensure Forward Secrecy, and re-signs the plaintext with a classical RSA Signature.
- **External Recipient side:** External legacy recipients receive the RSA-signed plaintext message via standard TLS. Replies are routed back to the Email Server, which leverages PiQASO to generate new PQC key materials and re-encrypts the response before routing it internally.

User Story Workflow

Sequence diagram description:

1. **Email Composition:** The Sender composes a 1-to-many broadcast email for external recipients. *(Note: UPKE Prerequisite is continuously handled asynchronously where the Email Server rotates its PQC Epoch Keys).*
2. **Routing Request:** The Sender Crypto Module requests routing keys for the external recipients from the PiQASO Server, sending their IDs.
3. **Coprocessor Encapsulation:** The PiQASO Server identifies the external targets as "Legacy" (No PQC support). It generates K_{msg} , encapsulates it with the Email Server's Epoch Key to create C_{key} , and returns $\{K_{msg}, C_{key}\}$ to the Sender.
4. **Local Encryption & Submission:** The Sender's module encrypts the payload with K_{msg} (AES-GCM) to create C_{msg} , destroys K_{msg} locally, and submits $\{C_{msg}, C_{key}\}$ to the Internal Email Server via standard TLS.
5. **Decapsulation Request:** The Internal Email Server requests the PiQASO Server to decapsulate C_{key} .
6. **Return Key:** The PiQASO Server decapsulates C_{key} into K_{msg} and returns it to the Email Server.
7. **Server-Side Decryption & Translation:** The Internal Email Server decrypts C_{msg} to Plaintext, permanently destroys K_{msg} to ensure Forward Secrecy, and applies a Classical RSA Signature.
8. **External Relay:** The Email Server relays the RSA-Signed Plaintext via standard external TLS to the External Legacy Servers.
9. **Inbox Delivery:** The external servers deliver to the external inboxes, where the recipients view the plaintext email (RSA Verified).
10. **Legacy Response:** The External Legacy Client sends a plaintext reply.

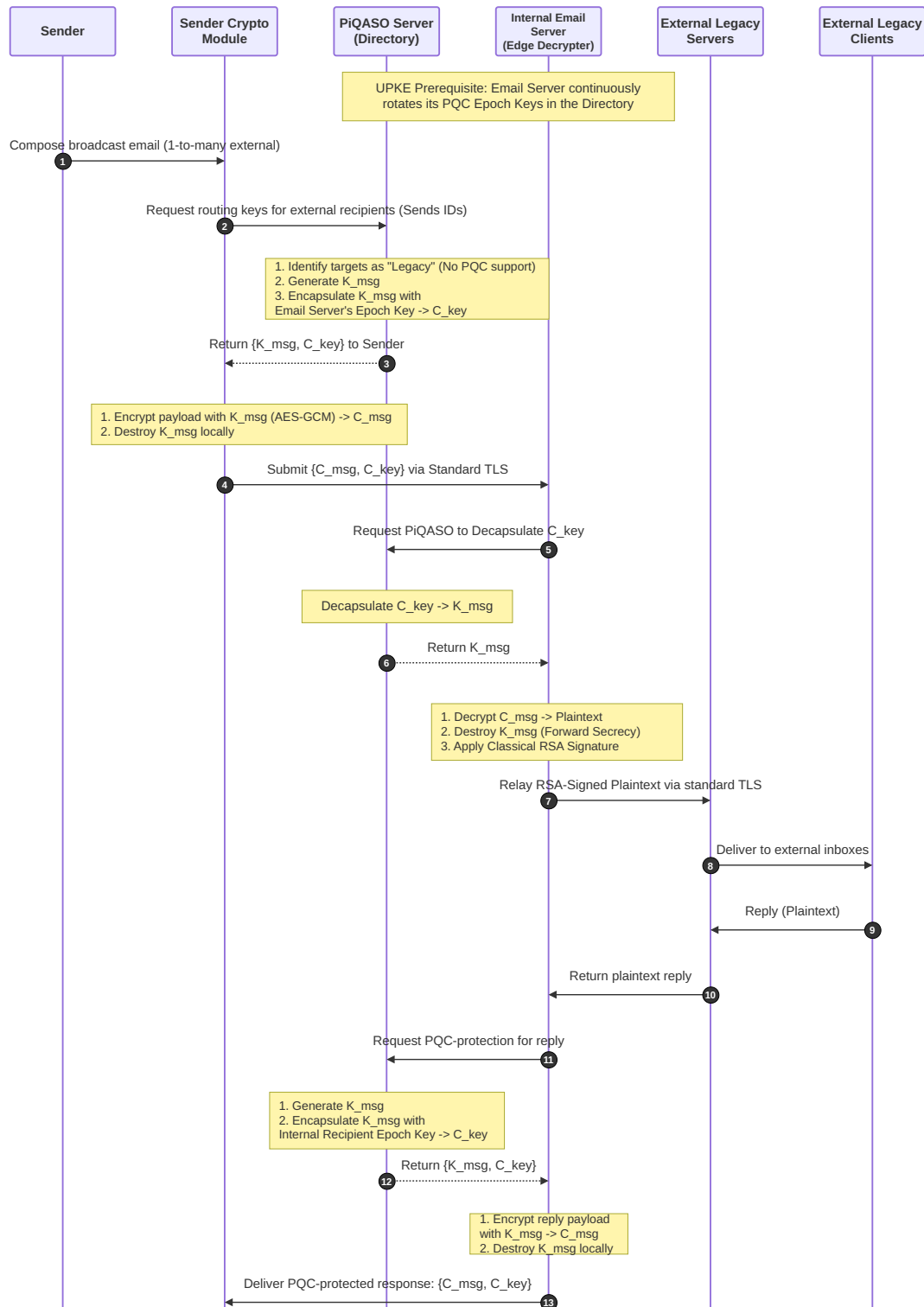


Figure 12.5: Sequence diagram for PQC-secured 1-to-many email exchange (Edge Decryption)

11. **Reply Routing:** The External Legacy Server returns the plaintext message to the Internal Email Server via standard TLS.
12. **Request Protection:** The Internal Email Server requests PQC-protection for the incoming reply from the PiQASO Server.

13. **PQC Re-protection & Delivery:** The PiQASO Server generates a new K_{msg} , encapsulates it with the internal recipient's Epoch Key into C_{key} , and returns them. The Email Server encrypts the reply payload with K_{msg} , destroys K_{msg} locally, and delivers the PQC-protected response $\{C_{msg}, C_{key}\}$ to the Sender Crypto Module.

Reference Scenarios

1. **Baseline (no PQC):** Standard TLS-based email exchange with MROs, leaving internal network routing vulnerable to future quantum interception ("Harvest Now, Decrypt Later") attacks.
2. **PQC-enabled Edge Decryption (Current State):** Secure 1-to-many distribution utilizing the Internal Email Server as a translation bridge and PiQASO as a cryptographic coprocessor, balancing strict internal PQC compliance with external legacy compatibility.
3. **PQC-enabled federated (Future State):** External relays and airline servers become natively PQC-compliant, enabling true end-to-end quantum protection and removing the need for server-side translation.
4. **Attack scenario:** A quantum-capable adversary attempts to intercept or forge communications; system resilience is measured by maintaining Forward Secrecy on the internal network and successful RSA verification on the external path.

12.7.3 Key Performance Indicators

Table 12.2: Application-specific KPIs for Secure Email Exchange Use Case

KPI	Description	Acceptance Criteria
Message Confidentiality Ratio	Proportion of transmitted and broadcasted messages that remain entirely secure and undecipherable to unauthorized third parties.	0% of PQC-encrypted messages are deciphered during simulated quantum-capable intercept attacks (100% confidentiality retention).
Authentication Accuracy	Proportion of incoming messages and responses verified as originating from a legitimate, untampered NEDHO collaborator or airline entity.	100% verification of genuine sender identities and 100% rejection/flagging of forged or altered emails.
Delivery Integrity Ratio	Proportion of broadcasted messages successfully received and correctly formatted by all intended collaborating airlines.	100% of target MRO/airline recipients successfully receive and can read the communication.
End-to-End PQC Latency Overhead	The additional time delay introduced into the email pipeline by PQC key generation, encapsulation, encryption, and decryption steps.	The average end-to-end processing and delivery overhead does not exceed 1.5 seconds compared to the classical TLS baseline.
Legacy Translation Success Rate	The operational efficiency of the PiQASO Gateway when translating messages between PQC environments and legacy external airline architectures.	100% successful execution of the software-based translation loop without message drops or cryptographic failures under normal operating loads.
Key Infrastructure Availability	The operational uptime and responsiveness of the central PiQASO Server during directory key provisioning requests.	The PiQASO Server achieves $\geq 99.9\%$ uptime and responds to key retrieval requests within 1s.

Chapter 13

PiQASO UC7 – Sector: Online Media

In the domain of online media and digital journalism, secure access management and strong user (employee) authentication is paramount to protecting the integrity and confidentiality of unreleased draft articles and sensitive intellectual properties. Current infrastructures primarily leverage PKIs relying on traditional RSA or Elliptic Curve Cryptography (ECC) which are completely vulnerable to quantum adversaries - meaning that a *“compromised network tap could lead to the exposure of confidential media archives tomorrow”*. Furthermore, TLS (as has been also the case for the entire application space considered in PiQASO) is the de-facto standard for exposing high-end security capabilities in user authentication and establishment of end-to-end secure channels over TCP connections, including bidirectional authentication, data confidentiality, and message integrity. Because of its flexibility in terms of cryptographic algorithms, it is also a popular choice in embedded environments; e.e., handheld devices used by journalists for the more flexible content management. Traditionally, such devices act as clients that connect to a backend Content Management System enabling for secure sessions with designated editors. Transitioning online media environments to quantum-resistant access management architectures ensures that content distribution and administrative workflows remain structurally sound and impenetrable to future quantum attacks. However, this is a rather challenging task due to the complexity posed of the multiple user roles that are part of such an ecosystem.

Primarily, such a PQC-enabled media platform requires a highly granular, role-based key encapsulation framework to govern content submission and editorial workflows. What this translates to is a complex key management system, that allows for **secure data processing** - robust computation and processing capabilities for executing complex data-sharing agreements with strict security and privacy controls. When an author uploads an unreleased article, the system must support complex multi-key orchestration: the content must be encrypted such that it can only be decrypted and reviewed by “editors” possessing the specific role cryptographic keys designated for that section. This demands a seamless key management system capable of mapping cryptographic identities to organizational roles, allowing secure multi-recipient encapsulation—such as generating distinct quantum-safe ciphertext tokens or using specialized key encapsulation mechanisms (KEMs)—ensuring data is strictly isolated to authorized reviewers without requiring a massive, latency-inducing overhead on the media host platform.

In this context, what PiQASO can instrument is such a key management system exposed through its “PQCaaS” environment allowing **client-based encryption/decryption of the article data payloads, while it exposes the core innovation of time-evolving key management as part of Updatable Public key Encryption/Attribute-based Encryption [55, 56] functionalities**, maintaining key chains over the time epochs (including generation of key update packages) per each user/client and attribute/policy. An Identity Manager component undertakes the user/device onboarding authentication and attribute-based authorisation actions.

13.1 Use Case's Description, Goals and Objectives

This use case addresses the secure, quantum-resistant operation of the editorial content pipeline of **DPG Media**. In the target workflow, journalists (authors) draft articles and reports and submit them through DPG Media's **Content Management System (CMS)** to its main publishing platform, where a designated editor or a designated set of editors within an editorial team reviews, approves and ultimately publishes the content on DPG Media's public-facing portals. The pipeline therefore spans three logical interactions:

1. author and editor *authentication* and access to the CMS,
2. *confidential submission* of an article from an author to the specific editor responsible for its review, and
3. *editorial review and publication* of the approved content.

Within PiQASO, the goal is **to make this end-to-end editorial pipeline resilient against the cybersecurity threats posed by quantum computers**, while at the same time raising the privacy and confidentiality guarantees of the editorial process well above the level offered by today's deployment. The use case pursues three objectives:

1. it upgrades the authentication of all roles that access the CMS (journalists, editors, editorial teams) to a **post-quantum, signature-authenticated secure channel**;
2. it protects each submitted article under a key that is specific to the editor entitled to review it, so that *confidentiality, forward and backward secrecy*, and *unlinkability* of editorial content are guaranteed;
3. it provides an optional, **attribute-based sharing** mode for content that has to be made available to a group of editors according to a policy rather than to a single named reviewer.

The cryptographic capability that anchors the second objective is **Updatable Public-Key Encryption (UPKE)**, delivered in this initial version as **Updatable Hybrid Encryption (UHE)**. The motivation is the following. Beyond the post-quantum authentication that protects a user when logging in to upload an article or a report, an article will frequently need to be reviewed by one specific editor, and by that editor only, or by one specific set of editors rather than by everyone with access to the platform.

The author therefore uploads content that is encrypted in such a way that only the intended reviewer can open it. Crucially, this is achieved not through attribute-based encryption but by encrypting each article under a key that is freshly derived for the target editor (or editor role) for a given *epoch*. Because every article is encrypted under a different key, the compromise of one key (and thus of one already-uploaded article) has no impact on the keys protecting any other article, whether previously or subsequently submitted. This yields both **forward secrecy** (compromise of current key material does not expose past content) and **backward secrecy** (it does not expose future content), and in combination they provide the *everlasting secrecy and privacy* that the editorial process requires.

The epoch is configurable to match operational practice: it can be defined per day, or per upload session in which a new article is submitted forward and backward secrecy hold across whichever epoch or session granularity DPG Media chooses. The underlying logic is that an editor holds a single **long-term private key**, from which multiple **per-epoch public keys** are derived. Each author uses one of these public keys to encrypt the articles that the editor will review, and every such key belongs to a different epoch, so that the compromise of one does not affect any other. A second privacy property follows directly from the construction: **unlinkability**. A journalist who reports on a sensitive topic may produce a series of

related articles over time. The scheme ensures that the set of articles, and the topics, that a given author has submitted for review cannot be linked together by any party other than the intended editor. Only the reviewing editor is able to link the submissions; no third party can correlate what a particular author has written. This unlinkability is what allows DPG Media to claim *privacy-preserving data sharing* of articles and to strengthen its compliance posture with respect to data-protection obligations such as the **GDPR**.

There is a clear need to migrate this workflow to **post-quantum cryptography**. Editorial archives and source material have long lifecycles and high sensitivity, and an adversary can capture protected traffic today and attempt to decrypt it once a *cryptographically relevant quantum computer* becomes available. As the public-key mechanisms that underpin authentication, key exchange and secure channels become vulnerable, preserving the confidentiality, authenticity and unlinkability of editorial content over long time horizons requires a transition plan that anticipates *cryptographic agility* and *long-term protection*. PiQASO addresses this need by combining post-quantum secure channels (**PQ-TLS**), post-quantum signatures and key encapsulation (**ML-DSA**, **ML-KEM**), and a post-quantum, updatable, cryptography-enforced sharing mechanism.

13.2 PiQASO Functionalities

Table 13.1: PiQASO Functionalities to be evaluated for Use Case 7 (Online Media)

PiQASO Functionality	Description and Purpose in the Use Case	RP1	RP2
PQ-TLS (incorporating ML-KEM and ML-DSA)	Establishes the quantum-safe CMS session for journalists, editors, and editorial teams, combining ML-KEM key encapsulation with ML-DSA-based authentication of the user's long-term identity certificate.	✓	
PQ Signatures (ML-DSA)	Used to cryptographically attest the OAuth-alike bearer token issued by the PQCaaS, ensuring CMS authorization requests are signature-authenticated and resistant to impersonation.	✓	
QR-TCB	Secure storage of cryptographic keys and supporting the execution of PiQASO PQC enablers.	✓	✓
PQCaaS	Mediates PQ-TLS authentication and CMS authorization for constrained/legacy clients, and hosts per-epoch ML-KEM key generation, encapsulation, and decapsulation for the article-sharing workflow.	✓	✓
UPKE (Updatable Hybrid Encryption, trusted-server mode)	Protects each submitted article under a fresh per-epoch, editor-specific key generated, encapsulated, and decapsulated by the trusted PQCaaS, delivering confidentiality, forward/backward secrecy, and unlinkability of submissions.	✓	
UPKE (fully lattice-based, reduced server trust)	Extends the UHE construction so that key generation, encapsulation, and decapsulation can also be performed at the journalist and editor clients, allowing the shared key to be derived without the server being trusted with it.		✓
ABE	Exploratory investigation into policy-based article sharing, encrypting content against an attribute-based access policy so that any editor with satisfying attributes can decrypt it.		✓

13.3 Components and Roles Considered

The UC7 scenario involves the following actors, components and systems:

- **Admin** :Configures the system's authentication (e.g., setting usernames and passwords) and establishes the core policies. These policies define the grouping of roles and determine which editorial teams have the specific context and authorization to review uploaded articles.
- **Journalist / Author**: Drafts articles and reports, submitting them to the CMS for review. The author currently uploads the content through the platform from a workstation or mobile device.
- **Editor / Reviewer** : A specific editor or a member of an editorial team responsible for reviewing and approving an author's submission. They are the designated party intended to read and validate the batched content
- **Editorial Team**: The organizational grouping that owns the review responsibility. The editorial team maintains the list of journalists whose submissions it must review and, when an upload is requested, designates the specific reviewer for that batch.
- **Data model**: The primary asset is an article: a composition of text and images, represented as a string/document. The content types handled are text, PDF, image and similar document formats there is no streaming in this environment. The pipeline is data-batching, not stream-based. An author uploads a complete article, which enters a review queue and only becomes available for publication after it has been reviewed and approved.

13.4 "As-Is" Scenario

In the current "**As-is**" setup, journalists and editors access DPG Media's CMS over a *conventional TLS-protected web session*, authenticating with predefined credentials (typically a username and password, optionally complemented by a second factor). Once authenticated, a journalist uploads an article to the CMS. The content is transmitted over classical TLS, stored in the DPG backend and placed into a queuing system. When an appropriate editor becomes available, they are matched with the queued article and notified to access it for review, subsequently providing their confirmation before it is published on the group's public-facing portals.

Security in this baseline relies on *conventional, classically secure primitives*: TLS for user-based authentication, classical signatures, and AES-based encryption for data at rest. Three structural limitations follow.

1. First, the system does not support **server-based authentication** over classical TLS, relying exclusively on user-based authentication. Consequently, channel establishment merely verifies that a user presented valid credentials. Furthermore, authentication currently depends on short-term user identities issued directly by the DPG Media CMS identity management system, meaning that the **long-term identity certificates** of users are not cryptographically authenticated as part of channel establishment.
2. Second, the DPG CMS lacks an inherent **Key Management System (KMS)**. Uploaded files are encrypted using a common key rather than per-article keys, meaning there is **no forward secrecy**. Consequently, if this common key or the CMS itself is compromised, it exposes not only all past and future articles, but also the identities of all authors and the records of who reviewed which article. Because there is no granular key protection, the loss of a single key effectively compromises the entire system.
3. Third, and most importantly for an editorial organisation, there is an inherent lack of **privacy protection** and **unlinkability**. Currently, there is an issue of *over-observation* among the editors and

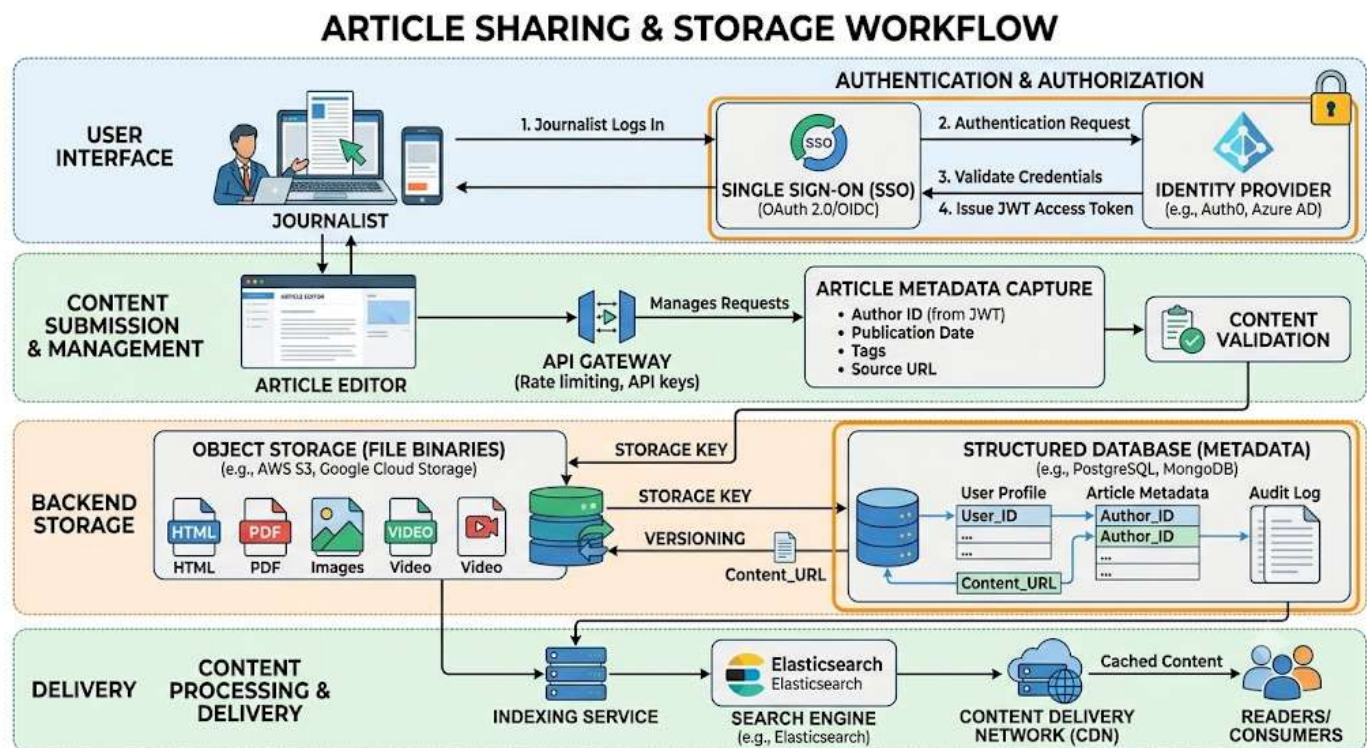


Figure 13.1: Article Sharing and Storage Workflow

authors who upload and review articles. This absence of privacy means that if the organisation were to offload this service to a third-party provider, that external system would have full visibility into all activities, data, and author-reviewer relationships. Consequently, there is no **data sovereignty** for the users; they are forced to rely entirely on the central platform. This structural reliance has direct consequences for *source protection* and data-protection compliance (e.g., GDPR). Additionally, all of these channels and stored assets rely on *classical public-key cryptography*, exposing them to **store-now-decrypt-later** attacks against long-lived, sensitive editorial material.

13.5 Challenges and Needs from PiQASO

The UC7 environment introduces technical and operational challenges that PiQASO must address to enable a realistic transition of the editorial pipeline to **post-quantum security**.

1. A primary challenge lies in establishing **authenticated, quantum-safe access** to the CMS across a heterogeneous ecosystem of roles and devices, ensuring a robust cryptographic binding between the user's *long-term identity* and the **Certificate Authority (CA)**. The authentication paradigm must transcend the mere establishment of a post-quantum-secure channel; it must rigorously authenticate these long-term identity certificates natively during channel establishment. Consequently, PIQASO must expose the architectural capability to integrate advanced **PQ-TLS** variants equipped with the embedded properties requisite for executing *signature-based authentication* of the user's long-term identity. Furthermore, this cryptographic capability must seamlessly interface with the platform's existing authorization frameworks, facilitating the issuance and validation of access tokens that are themselves inherently fortified with **post-quantum signatures**.
2. A second, pivotal challenge involves the design and implementation of an advanced **Key Management System (KMS)** tailored for cloud-centric information architectures, such as the DPG

Media CMS. This KMS must transcend baseline quantum security; it must intrinsically leverage post-quantum cryptography to enforce strict **confidentiality**, robust **forward secrecy**, and mutual **unlinkability**. To effectively mitigate the impact of any potential compromise, the system necessitates configurable *key rotation* mechanisms that partition cryptographic lifecycles into discrete *epochs*. This ensures that the exposure of a single key cannot jeopardize historical or future submissions. Functionally, this challenge is initially resolved through the deployment of the first iteration of our updatable-encryption capability, provisioned as **Updateable Hybrid Encryption (UHE)**.

3. A third critical challenge centers on **data sovereignty**, the reduction of *centralized trust assumptions*, and robust support for constrained clients. While the architecture must permit remote edge clients to seamlessly offload computationally intensive operations to the **PQCaaS** instantiated in the DPG backend, it is imperative to progressively diminish the trust placed in the central CMS. To establish authentic data sovereignty, PIQASO must empower users with the autonomous capability to generate and govern their own cryptographic keys. This strategic objective is realized during the second reporting period through the transition to a comprehensive, fully lattice-based **Updatable Public-Key Encryption (UPKE)** framework. Ultimately, this advanced UPKE construction resolves the third challenge, maximizing user sovereignty while minimizing central platform reliance, while simultaneously satisfying the stringent key management and forward secrecy requirements delineated in the second challenge.

13.6 "To-Be" Reference Scenario #1: Quantum-Safe Authentication and CMS Access

In the primary reference scenario, PIQASO comprehensively secures the **authentication and CMS-access pathways** across all editorial roles. The protocol proceeds as follows:

1. **Session initialization.** Upon system initialization by a journalist, editor, or editorial team, the client establishes an enhanced **PQ-TLS** session with the DPG backend. This session is negotiated either locally via the PIQASO PQC SDK or, for constrained edge devices (e.g., mobile platforms), via the centralized **PQCaaS**, mirroring the authentication paradigms utilized in other PIQASO use cases.
2. **Legacy interoperability (optional path).** To ensure architectural interoperability, users may optimally authenticate via the legacy portal using classical TLS; this portal subsequently acts as a secure proxy, forwarding the connection to the PQCaaS over PQ-TLS.
3. **Cryptographic handshake.** The handshake fundamentally relies upon the extended PIQASO protocol, synergistically leveraging **ML-KEM** key pairs for secure encapsulation alongside **ML-DSA** for the *signature-based authentication* of the user's long-term identity. This critical enhancement addresses a **structural vulnerability** in standard PQ-TLS implementations, which do not natively cryptographically authenticate long-term identity certificates.
4. **OAuth-alike session authorization.** Following the establishment of the PQ-TLS session, the architecture emulates the *OAuth authorization framework* to orchestrate CMS access. The PQCaaS initializes a session and generates an OAuth-alike bearer token that cryptographically embeds the user's IP address.
5. **Surrogate authentication request.** Utilizing this OAuth-alike flow, the PQCaaS executes a surrogate authentication request against the database on behalf of the user.

6. **Token attestation.** Should the database lack an out-of-band communication channel to verify the request directly with the user, the PQCaaS cryptographically attests to the token by signing it with **ML-DSA** and transmits it back to the client device.
7. **CMS access authorization.** To gain frontend CMS access, the device presents this ML-DSA-signed OAuth-alike bearer token to authorize the connection.

Ultimately, this comprehensive scenario guarantees **quantum-safe, signature-authenticated CMS access** across all roles. It systematically deprecates the reliance on classical TLS and credential-only authentication, thereby establishing the robust cryptographic foundation upon which the privacy-preserving article-sharing mechanisms of Reference Scenario #2 operate.

13.7 "To-Be" Reference Scenario #2: Privacy-Preserving Article Sharing with Updatable (Hybrid) Public-Key Encryption

The second reference scenario delivers the distinctive capability of the use case: leveraging **Updatable Public-Key Encryption / Updatable Hybrid Encryption** for **privacy-preserving sharing of articles**, so that an article submitted by an author is confidential to its intended reviewer and is, in addition, **unlinkable**.

The participants are the server, the sender (the journalist/author) and the receiver (the editor who reviews the submission). After a successful PQ-TLS authentication (Reference Scenario #1), the editorial team sends the PQCaaS the list of journalists under its review responsibility, and the PQCaaS uses this list to associate each journalist with the editorial team responsible for reviewing their submissions. The secure session established in Reference Scenario #1 carries all of these exchanges.

The protocol proceeds as follows:

1. **Per-epoch key establishment.** Mediated by the PQCaaS, the reviewer (editor) sends its identity to the server and requests a public/private key pair. The PQCaaS takes the receiver's identity, initializes the current *epoch*, and invokes **ML-KEM** key generation to create the editor's per-epoch private/public key pair. It returns the editor's private/public key material to the editor over PQ-TLS and forwards the editor's per-epoch public key to the journalist.
2. **Encrypted upload.** When the journalist wishes to upload an article, it signals the event "I want to upload an article" to the PQCaaS, which pings the corresponding editorial team and receiver. The designated editor triggers the establishment of the per-epoch key. The journalist then asks the server to create a shared secret for the target editor: the PQCaaS invokes **ML-KEM encapsulation** using the reviewer's per-epoch public key, producing a shared secret K and the associated kem-ct. The server returns the kem-ct and the key K to the journalist over PQ-TLS. The journalist encrypts the article under K and sends the resulting ciphertext through the platform to the reviewer.
3. **Decryption and review.** The reviewer requests the relevant epoch and the public key of the journalist who submitted the ciphertext, and invokes **ML-KEM decapsulation** using its per-epoch private key on the received kem-ct, reconstructing exactly the same shared secret K . With K the editor decrypts and reviews the article.
4. **Epoch update.** Finally, the PQCaaS updates the epoch, deriving a fresh per-epoch private/public key pair for the editor so that the next submission is protected under new key material.

Because the editor relies on the PQCaaS (hybrid encryption) to produce a different ML-KEM key pair per epoch, and because these per-epoch public keys (combined, through encapsulation and decapsulation, with the journalist's key material) yield a fresh symmetric key K for every submission, the protected articles are mutually **unlinkable**: only the editor can link the submissions it is entitled to review, and in this initial version the unlinkability operation can be performed only by the (trusted) server. The construction thereby delivers **confidentiality** scoped to the intended reviewer, **forward and backward secrecy** across epochs, and **unlinkability**. *The everlasting secrecy and privacy that motivate the use case*, with the corresponding benefit for *source protection* and **GDPR compliance**.

This scenario defines two modes of operation:

1. **Initial version (M18)**. The server (PQCaaS) is *fully trusted*, as it manages the editor key material and performs key generation, encapsulation and decapsulation; this is the mode demonstrated in M18.
2. **Fully decentralized version (M36)**. This trust assumption is **weakened**: key generation, encapsulation and decapsulation can also be performed at the client and at the receiver, so that the shared key K is derived independently at the journalist and at the editor **without the server having to be trusted with it**. The fully UPKE construction underlying this second mode is developed jointly with the project's cryptography partners and is delivered in M36.

As a prospective, forward-looking extension to the core UPKE paradigm, the PIQASO project investigates the theoretical applicability of **Attribute-Based Encryption (ABE)**, Figure 3.4. It must be explicitly noted that this ABE variant constitutes an *exploratory architectural concept* rather than a strictly committed deliverable for the current reporting periods. While UPKE enforces rigorous privacy scoped to statically designated reviewers and cryptographic epochs, ABE introduces a fundamental paradigm shift: transitioning from static, identity-bound role definitions to **dynamic, fine-grained, attribute-driven access control**. Under this theoretical construct, confidentiality is achieved by encrypting an article payload against an expressive access policy comprising specific professional attributes. Consequently, any editor in possession of a cryptographic private key whose embedded attributes mathematically satisfy this policy is intrinsically authorized to decrypt and evaluate the content. Given the conceptual alignment between ABE and UPKE in advancing confidentiality and data sovereignty, this optional investigation provides a strategic blueprint for future editorial environments that demand highly elastic, policy-based review mechanisms independent of predefined organizational silos.

13.8 Reference Scenario User Stories and Metrics of Success

13.8.1 User Story #1 (Reference Scenario #1): Quantum-Safe Authentication and CMS Access (Journalist / Editor)

UC7.US1: Post-Quantum-Secure CMS Login and Identity Authentication

As a Journalist or Editor, I want to log in to the CMS so that my session is established over a post-quantum-secure channel and my long-term identity certificate is cryptographically authenticated so that impersonation is prevented even against a future quantum-capable attacker.

User Story Confirmation

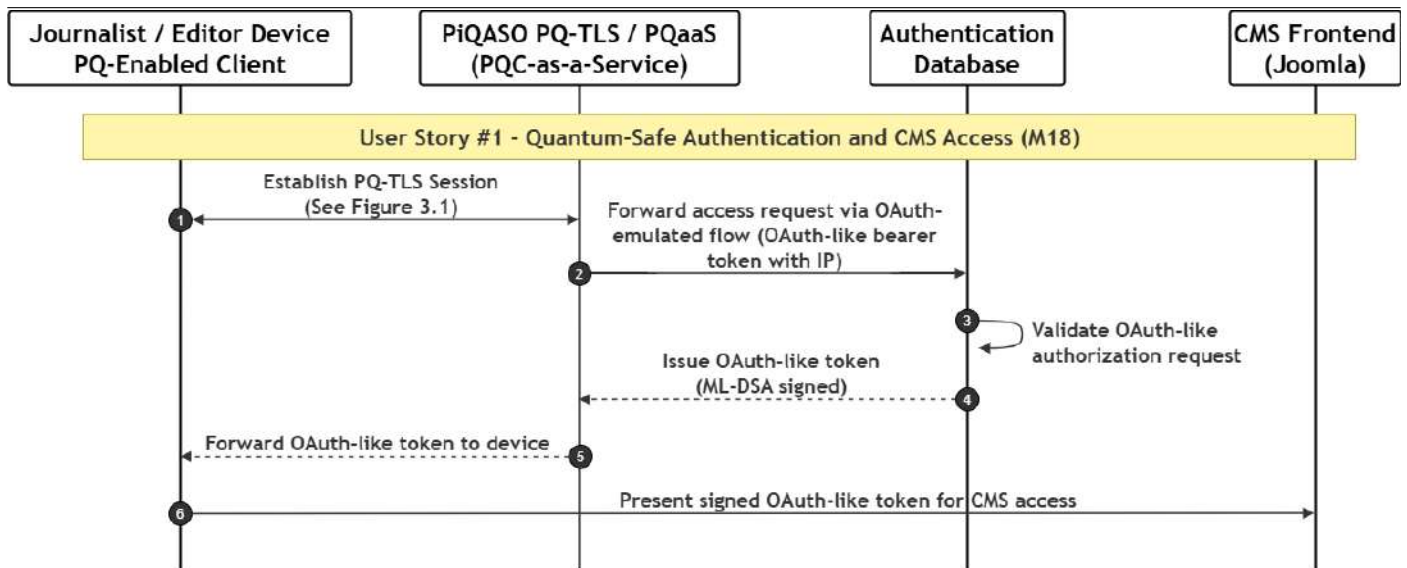


Figure 13.2: Sequence diagram for Quantum-Safe Authentication and CMS Access

-The CMS session is established over a post-quantum-secure channel: the handshake inherently leverages ML-KEM for robust key establishment and ML-DSA for the signature-based authentication of the user's long-term identity.

-The long-term identity certificate of the journalist or editor is cryptographically authenticated via ML-DSA, mathematically precluding user impersonation, even against future quantum-capable adversaries lacking the corresponding private key material.

-Constrained devices incapable of executing local PQC operations are seamlessly authenticated via the PQCaaS, meticulously preserving the channel's post-quantum security guarantees. Furthermore, legacy portals utilizing classical TLS are securely proxied to the PQ-TLS infrastructure via the PQCaaS.

-CMS authorization is governed by an OAuth-alike bearer token that is rigorously signed with ML-DSA; this token is systematically validated during each subsequent access request to the CMS frontend.

User Story Workflows

Sequence diagram description:

1. The client device establishes a PQ-TLS session with the PQCaaS, utilizing either the local PiQASO PQC SDK or the centralized PQCaaS for constrained edge devices. Alternatively, users may connect to the legacy portal via classical TLS, which subsequently forwards the secure connection to the PQCaaS via PQ-TLS. This handshake explicitly executes the extended PiQASO protocol, leveraging both ML-KEM key pairs and ML-DSA to securely authenticate the long-term identity.
2. The PQCaaS initializes a session, generates a stateful token (embedding the user's IP address), and forwards the CMS access request to the Authentication Database, thereby emulating the OAuth authorization process on behalf of the user.
3. The Authentication Database receives the redirection and processes the authorization request utilizing the OAuth-alike bearer token framework.
4. In the absence of an out-of-band verification mechanism, the token is cryptographically attested, issuing an ML-DSA-signed OAuth-alike bearer token.
5. The PQCaaS securely forwards this ML-DSA-signed OAuth-alike bearer token back to the user's client device.

6. For all subsequent interactions, the client device accesses the CMS frontend by presenting this ML-DSA-signed OAuth-alike bearer token to seamlessly and securely authenticate against the database.

13.8.2 User Story #2 (Reference Scenario #1): Privacy-Preserving Article Upload and Review (UHE/UPKE)

UC7.US2: Confidential and Unlinkable Article Submission

As a Journalist, I want to upload an article that is readable only by the designated editor and protected under a fresh per-epoch key so that my submission remains confidential to unauthorized parties, and the set of articles and topics I submit cannot be linked together by anyone other than that editor.

User Story Confirmation

- Each article is encrypted under a per-epoch, editor-specific key derived via UHE/UPKE; a different key protects every submission/epoch.
- The compromise of one article's key does not expose any other (past or future) submission (forward and backward secrecy).
- Only the intended editor can decrypt the submission and link the author's submissions (unlinkability); in the UHE version, the operation is mediated only by the trusted PQaaS.

User Story Workflows

Sequence diagram description:

1. The editor sends its identity to the server and requests a public-private key pair.
2. The PQaaS initiates the epoch and runs ML-KEM key generation, producing the editor's public key pair (sk, pk).
3. The PQaaS returns the editor's ML-KEM (sk, pk) over PQ-TLS.
4. The PQaaS forwards the editor's per-epoch public key pk to the journalist.
5. In the upload phase, the journalist asks the PQaaS, for editor X, to create a shared secret (request encapsulation).
6. The PQaaS runs ML-KEM encapsulation with pk, producing the shared secret K and the kem-ct.
7. PQ-TLS authentication.
8. The PQaaS sends the kem-ct and the key K to the journalist over PQ-TLS.
9. The journalist encrypts the article with K (hybrid symmetric encryption).
10. The journalist sends the ciphertext to the editor via the platform.
11. Extract info on KEM function based on credential.
12. The editor requests the epoch and the kem-ct from the PQaaS (for decapsulation).
13. The PQaaS runs ML-KEM decapsulation with the editor's SK, reconstructing the same K.

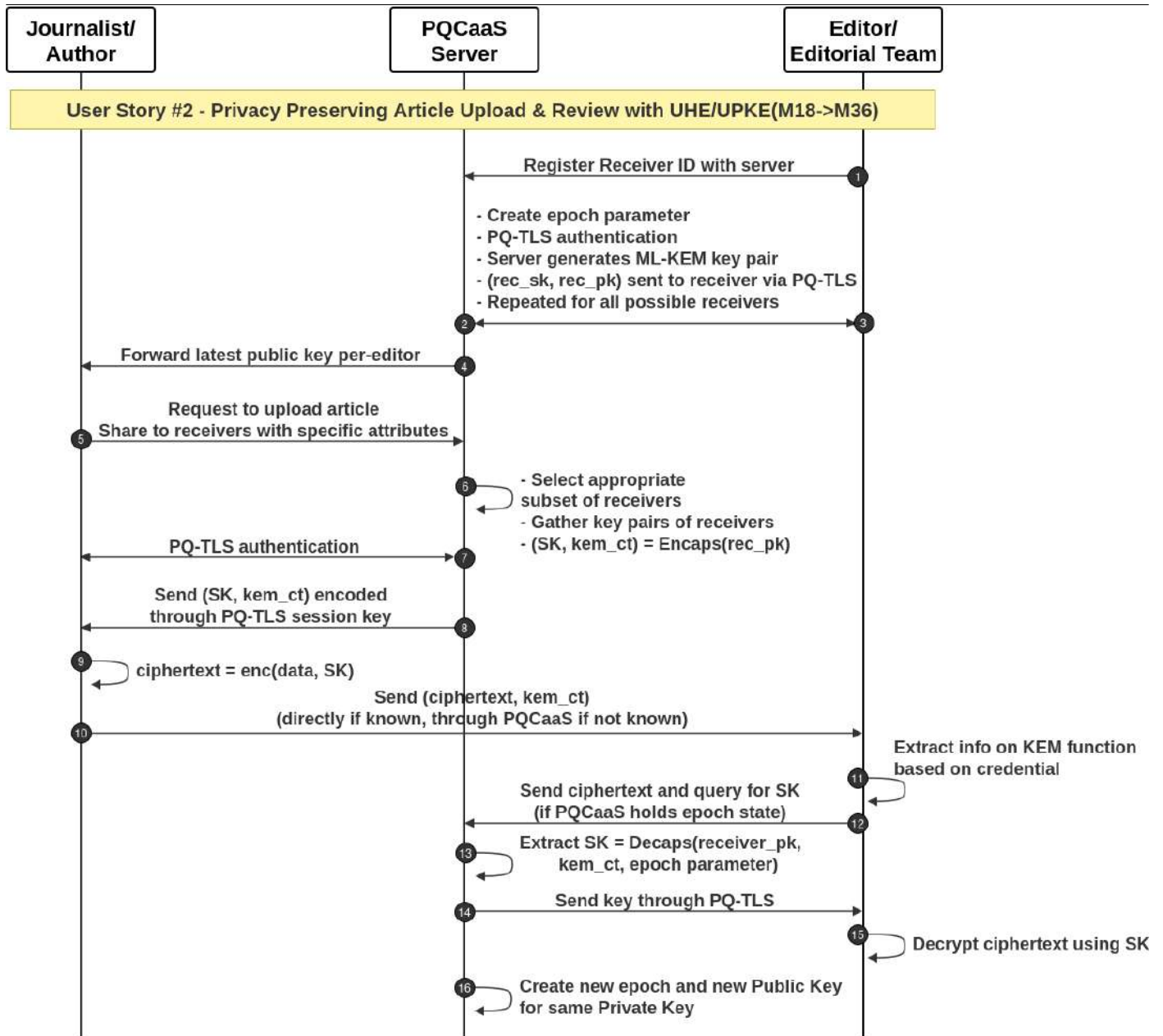


Figure 13.3: Sequence diagram for Privacy-Preserving Article Upload and Review (UHE/UPKE)

14. The PQaaS sends K back to the editor.
15. The editor decrypts the article with K and reviews it.
16. Upon receiving an explicit notification from the editor, the PQaaS updates the cryptographic epoch, deriving a fresh per-epoch key pair $(sk + 1, pk + 1)$ for the editor.

Sequence diagram description:

1. The journalist defines a dynamic access policy and requests the generation of ABE primitives.
2. The PQaaS set up ABE (see Figure 3.4 for protocol-level details).
3. The PQaaS generate ML-KEM keypair and invoke ABE.

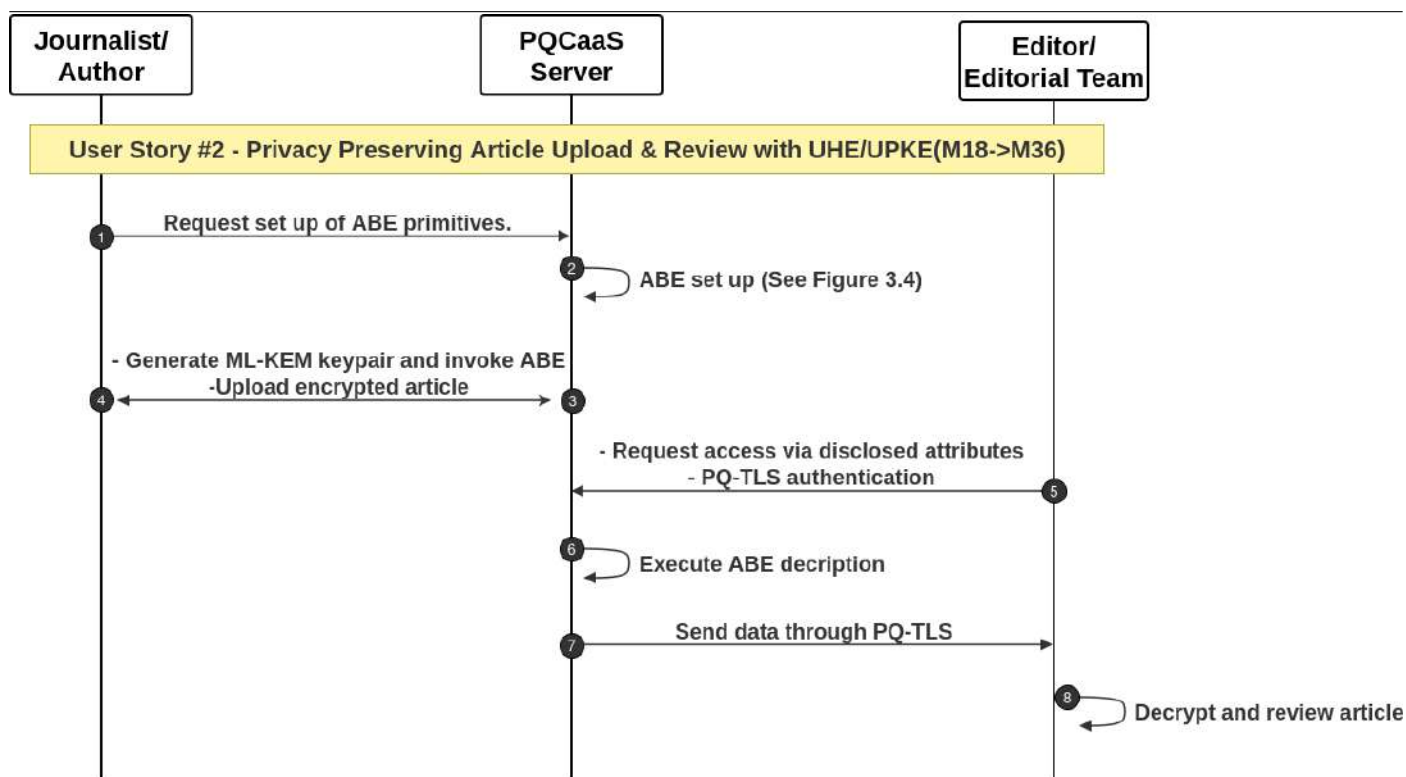


Figure 13.4: Sequence diagram for Privacy-Preserving Article Upload and Review (ABE)

4. The journalist encrypts the article with K (hybrid symmetric encryption) and uploads the encrypted batch.
5. An editor requests access to the article via disclosed attributes to the PQCaaS for authentication. PQ-TLS authentication.
6. The PQCaaS evaluates the editor's attributes against policy P . Upon a successful match, it executes ABE decryption .
7. The PQCaaS securely provisions the reconstructed symmetric key K to the authorized editor.
8. The editor utilizes K to decrypt the article payload and comprehensively review the submission.

13.8.3 Key Performance Indicators

Table 13.2: Application-specific KPIs

KPI	Description	Acceptance Criteria
Seamless User Experience	Users (journalists) should experience zero prompts asking them to re-upload or re-encrypt old articles because a reviewer left the department or a key expired	Zero Re-encryption Cycles and Transparent Key Derivation. The generation of the keys must happen transparently in the background without any disruption to the user data upload operation.

End-to-End PQC Latency Overhead	The cumulative processing delay that the post-quantum operations of the pipeline add to a login-and-submission cycle, measured against a classical TLS reference.	The entire process of user authentication; time epoch key establishment and update; data payload encryption/decryption; message integrity verification should not account for more than 45% of the overall time required for the secure upload and review of an unpublished document.
PQCaaS Availability	The service availability and responsiveness of the central PQCaaS as it handles authentication, token issuance and per-epoch key provisioning requests.	$O(1)$ scaling. As the number of users for authentication scale up as well as the number of attributes or roles assigned to an editor, the verification process of the PQCaaS must scale linearly to the attribute design space.
Authentication Latency	Time required for the two communicating parties to establish an authenticated encrypted communication channel. Experimentation of PiQASO updated handshake profiles/templates enabling the “ <i>hybridization</i> ” of the key encapsulation process through adding the supported KEM algorithms in the handshake message exchange [60]	Authentication (including key generation and key establishment) should be able to execute in less than 660ms; Testing also for scalability extending the authentication capabilities of the PQCaaS to be able to manage more than 500 users/clients. Focus here is on the detailed benchmarking between a legacy TLS 1.3 handshake and the PiQASO PQ-TLS functionality.
Handshake Fragmentation & Congestion Window	Tracking if the large public credential size (PQ-encoded certificates) forces the connection into multiple Round-Trip times (RTTs) requiring from the user client to “wait” for out-of-order packets containing parts of the signature prior to the establishment of an authenticated encrypted communication channel	Packet fragmentation of certificate exchange ≤ 5 packets; RTT ≤ 45 ms (ensuring it falls within the human perception threshold of “instantaneous”)
Resource Consumption	Computational resources needed on the client-side for executing PQ-TLS and the advanced crypto construction for enabling the protection of exchanged data payloads through the Updatable Public Key Encryption (UPKE)	Memory Consumption ≤ 120 KB RAM (including also the management/storage of epoch keys in the case of an “untrusted” PQCaaS server [56]); End-to-End latency for the key update process between user (e.g., journalist) and receiver (i.e., editor) ≤ 220 ms accounting also for the network overhead posed; Time required to reconfigure the PiQASO underlying HW (acceleration) module from an ML-KEM step to an ML-DSA verification step (≤ 660 ms)

Key Management	Establishment, maintenance, and update of two-types of evolving key chains: user-specific and attribute/policy-specific key chains for allowing (i) Data sharing between a specific user (data receiver) and one or more senders utilizes the evolving user-specific key chains mediated by the PQ-CaaS server, and (ii) data sharing between a specific sender and multiple receivers complying with the CMS-defined policy.	Both modes of operation should complete in less than 550ms. Especially, for the latter case where all receiver authentication needs to also consider the verification of specific attributes, this process should not account for an additional overhead of higher than 300ms; Number of active role keys that can reside directly inside the fast local RAM caches of the PQCaaS and/or user rather than been offloaded to a file system requiring additional operations for the secure loading (≥ 20 keys and certificates).
-----------------------	--	---

Chapter 14

PiQASO UC8 – Sector: UAVs

14.1 Use case's Description, Goals and Objectives

In the evolving landscape of unmanned aerial systems (UAS), ensuring the security and integrity of communications between the ground control station (GCS) and the airborne platform has emerged as a paramount concern, especially in the context of quantum-era threats. Use Case 8, defined within the scope of the PiQASO project, addresses this challenge by designing, implementing, and validating a secure, post-quantum-resistant communication architecture for mission-critical UAV operations.

The operational scenario is built around the CERBERUS UAV platform, which is enhanced with a Post-Quantum Trusted Computing Base (PQ-TCB) deployed on an AMD Zynq UltraScale+ MPSoC ZCU104 board. This hardware not only supports the avionics stack but also accelerates cryptographic operations related to the PiQASO PQC SDK. The mission begins with a preparatory stage during which secure provisioning of cryptographic material is performed. Post-quantum credentials, encompassing Kyber-based key encapsulation and Dilithium-based digital signatures, are generated and securely stored either on the onboard platform or registered through the PQ-DS server in the PQCaaS configuration.

Once the cryptographic credentials have been provisioned, the secure channel establishment phase follows. Before take-off, the GCS and UAV perform a mutual authentication handshake using a PQ-TLS or KEM-TLS protocol. In the local SDK deployment, all cryptographic exchanges and verifications are processed directly on the ZCU104 module. In contrast, in the PQCaaS variant, cryptographic computations are securely delegated to the PQ-DS server, allowing the UAV and ground station to act as lightweight clients. This design permits scalability in scenarios involving multiple GCSs or UAV swarms, while maintaining a uniform trust architecture.

With the secure channel established, mission-critical data—including waypoints, geofencing parameters, and telemetry—is transmitted over authenticated and encrypted links. The use of AEAD schemes ensures the confidentiality and integrity of the command-and-control (C2) traffic throughout the flight. Notably, telemetry and command data benefit from symmetric keys derived during the initial PQC handshake, ensuring ongoing security without compromising link latency or bandwidth. Furthermore, the framework enables secure data storage at rest: mission logs and operational data are encrypted locally and can be offloaded to the PQ-DS for long-term secure storage and audit compliance.

A critical aspect of this implementation is the system's resilience in dynamic conditions. If the integrity of the communication tunnel is compromised, predefined contingency procedures are activated, such as return-to-home or controlled landing. These safety-driven behaviors tightly couple cybersecurity event detection with real-time flight logic. This linkage ensures that cyber events cannot silently degrade the mission but are instead mitigated in-flight with clear operational consequences.

Moreover, the architecture is designed with extensibility in mind. The same PQCaaS-based model sup-

ports swarming UAV configurations and operations involving multiple ground control stations. Each entity in the network—whether airborne or terrestrial—establishes its own independent PQ-TLS session with the PQ-DS server, ensuring a scalable and interoperable security posture across collaborative mission environments. This is particularly relevant in use cases requiring coordinated logistics, inspection, or emergency response operations involving several UAVs exchanging sensitive data over untrusted networks.

The use case delivers several engineering and operational advantages. It validates that PQC can be integrated into resource-constrained UAVs without degrading mission performance. Furthermore, it provides quantitative insights into the trade-offs between local processing (SDK) and remote cryptographic delegation (PQCaaS), thus informing future design decisions for UAV architectures. Most importantly, the UC8 scenario establishes a reusable blueprint for secure C2 in the quantum era, emphasizing secure authentication, tamper-proof data exchange, end-to-end confidentiality, and post-quantum resilience.

14.2 PiQASO Functionalities

Table 14.1: PiQASO Functionalities to be evaluated for Use Case 8 (UAVs)

PiQASO Functional-ity	Description and Purpose in the Use Case	RP1	RP2
PQ-TLS / KEM-TLS (incorporating ML-KEM)	Establishes the mutually authenticated, quantum-resistant control-plane session between the GCS and the UAV (Scenario 1, local SDK) and between each node and the PQCaaS/PQ-DES backend (Scenario 2), forming the arming precondition.	✓	✓
PQ Signatures (ML-DSA)	Provides signature-based mutual authentication of the GCS and UAV during handshake completion, and signs/verifies handshake artefacts to prevent impersonation, both locally (Scenario 1) and via PQCaaS (Scenario 2).	✓	✓
QR-TCB	Secure storage of cryptographic keys and supporting the execution of PiQASO PQC enablers.	✓	✓
Hardware Acceleration	Reduced-scope acceleration of selected ML-KEM/ML-DSA operations on the ZCU104 FPGA board, benchmarked against a software-only PQC SDK baseline to preserve mission responsiveness.	✓	
PQCaaS	Delegates selected ML-KEM, ML-DSA, credential-management, and session-establishment operations to the PQCaaS/PQ-DES backend, enabling lightweight UAV/GCS clients and centralised key governance for multi-ground-station and swarm scenarios.		✓
ABE	Encrypts selected UAV-generated data objects (e.g., imagery, sensor products, mission logs) under an access policy so that only ground/base stations with satisfying attributes can decrypt them, for the multi-ground-station data-sharing workflow.		✓

14.3 “As-is” Scenario

Prior to the integration of the PiQASO security framework in Use Case 8, the operational foundation of UAV missions followed a conventional command-and-control (C2) architecture, lacking mechanisms to ensure verifiable quantum-resistant communication security. The baseline setup relied on standard C2 links and unprotected mission upload procedures between the ground control station and the UAV. While some implementations attempted to embed basic cryptographic features for confidentiality, integrity,

or authentication, these were typically partial, inconsistent across platforms, or entirely dependent on classical cryptographic primitives vulnerable to emerging quantum threats.

This structural weakness becomes particularly apparent in scenarios involving “harvest now, decrypt later” adversarial models. In such cases, even encrypted mission data—if captured during transmission—could later be decrypted once quantum computing matures, revealing sensitive information long after the mission has concluded. Consequently, the as-is posture presents both immediate and long-term risks to mission confidentiality, system integrity, and overall operational assurance.

A typical mission scenario within this baseline context—such as one executed by the CERBERUS UAV platform—follows a standard operational flow. It begins with mission planning activities conducted at the ground control station, followed by pre-flight transmission of mission-critical parameters such as waypoints and flight constraints. The UAV is then armed and launched, after which it proceeds through a flight path dictated by uploaded instructions. During the flight, telemetry data and command packets are exchanged between the UAV and ground station, culminating in a final phase of landing and post-mission data extraction. Critically, this process often proceeds without any enforcement of secure or mutually authenticated communication channels.

The absence of a quantum-resilient security overlay in this architecture exposes the UAV-ground segment link to a wide spectrum of cyber threats. One of the most prevalent vulnerabilities is passive eavesdropping, which can result in the unauthorized extraction of telemetry data and mission parameters in real time. More advanced threats include active Man-in-the-Middle (MITM) attacks, in which malicious actors intercept and alter command packets, potentially gaining unauthorized control of the UAV or manipulating its behavior mid-flight.

Additionally, without the implementation of cryptographic anti-replay mechanisms, previously valid commands may be captured and resent by an adversary to induce unauthorized behavior. This replay risk is particularly acute in scenarios where command validation does not enforce strict temporal or contextual coupling. Moreover, if the ground control system is compromised—either through malware, misconfiguration, or other upstream vulnerabilities—malicious commands can be generated and transmitted even before the data enters the radio link, rendering downstream encryption ineffective.

These gaps underline the pressing need for a robust post-quantum cryptographic framework that enforces mutual authentication, prevents unauthorized access, and ensures communication integrity. Use Case 8 under PIQASO directly addresses these vulnerabilities through the deployment of PQ-TLS and KEM-TLS protocols, strong identity bindings, and cryptographically secured data exchange, both in local deployments and cloud-assisted (PQCaaS) modalities. This transition from the baseline scenario to a quantum-resilient operational posture forms the foundation of the PIQASO security transformation strategy.

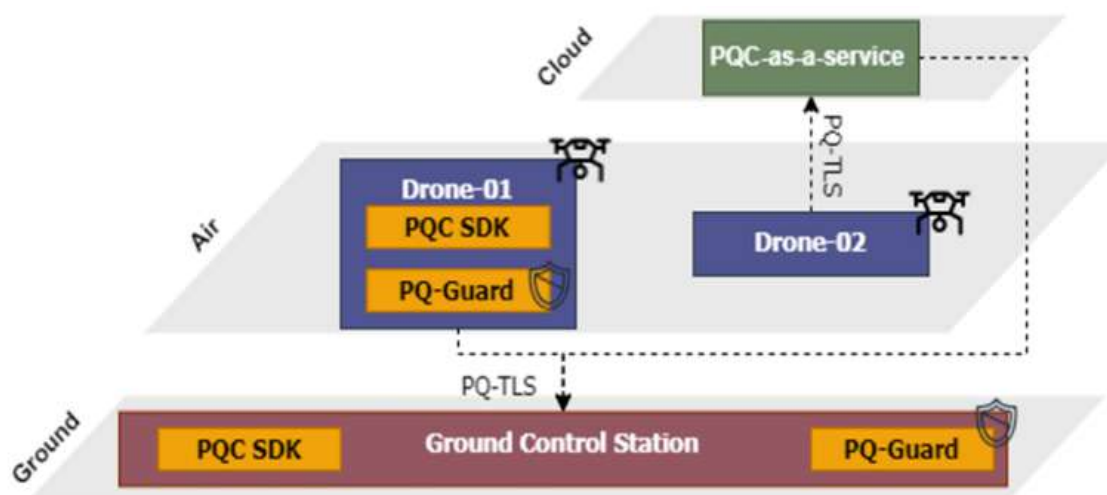


Figure 14.1: Figure 1 High level illustration of the use case 8

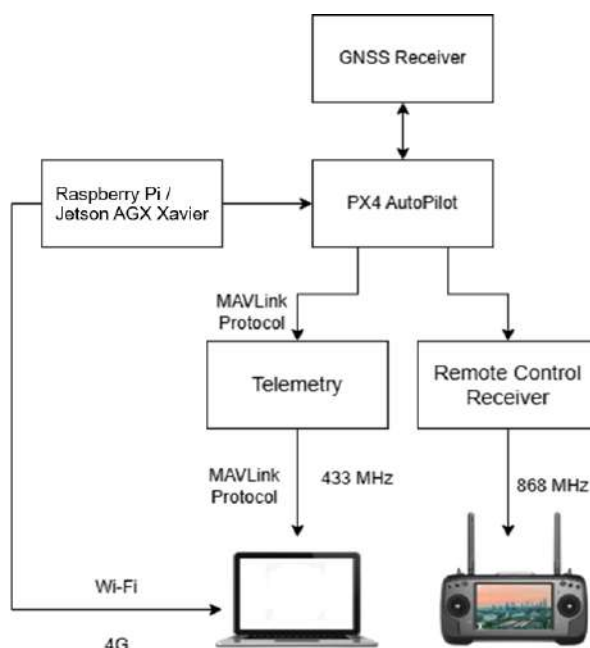


Figure 14.2: Figure 2 UAV as is high level architecture

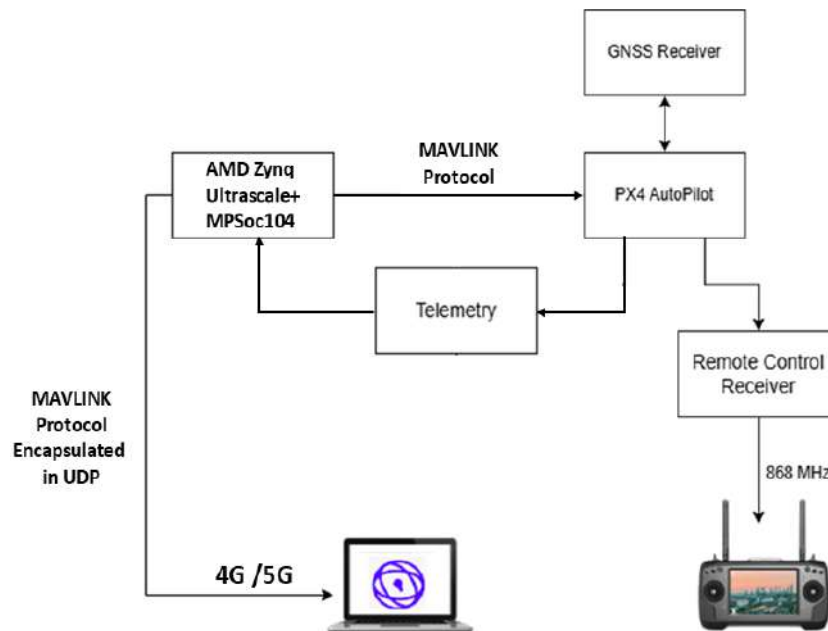


Figure 14.3: Figure 2 Piqaso High Level UAV Architecture

14.4 Components and roles considered in the context of UAV Use Case

The technical architecture of Use Case 8 (UC8) within the PiQASO framework reflects a rigorously defined division of operational responsibilities and a comprehensive stack of integrated technologies aimed at securing UAV command-and-control (C2) communications against both current and emerging threats. The architecture captures two operational modalities—local and service-based—each aligned with a distinct set of components, processes, and trust anchors, while supporting modularity, scalability, and robustness for mission-critical UAV deployments.

14.4.1 System model and common communication path

UC8 considers an end-to-end command-and-control communication chain comprising the Ground Control Station (GCS), the 5G/IP communication infrastructure, and the CERBERUS UAV. The airborne system includes the PX4 flight-control subsystem, the GNSS receiver, and a companion-computing and security platform based on the AMD Zynq UltraScale+ MPSoC ZCU104. In Reference Scenario 2, the system model additionally includes the PiQASO PQC-as-a-Service backend and its PQ Data Encryption Server.

Mission commands, waypoints, geofencing parameters and telemetry are generated and interpreted using MAVLink. In order to preserve compatibility with existing industrial UAV software and to ensure that the evaluation remains relevant to current UAV deployments, UC8 shall not redefine the MAVLink message format, alter its semantics, or introduce a UC8-specific MAVLink dialect. Instead, the original encoded MAVLink frames shall be encapsulated unchanged at application level into UDP datagrams and transported over the 5G/IP communication link.

The same MAVLink message set, message ordering, telemetry rates and packetisation policy shall be used for the as-is baseline and for both to-be reference scenarios. The application payload will therefore remain identical across the experiments, while the resulting on-wire packets will differ because of the cryptographic metadata, authentication tags, tunnel headers and other protection overhead introduced

by PiQASO. This controlled arrangement will allow the impact of PQC to be measured independently from changes to the flight-control protocol.

The selected PQ-TLS or KEM-TLS profile is used as the control-plane mechanism for mutual authentication and post-quantum session-key establishment. The real-time data plane remains MAVLink-over-UDP and is protected using authenticated encryption with associated data under the session keys established by the post-quantum handshake. This distinction prevents the system model from incorrectly implying that standard TLS directly transports UDP traffic.

Measurement points shall be introduced at both endpoints to record: the time required to encode and decode MAVLink frames; the size of the unprotected and protected packets; the time required to protect and unprotect traffic; end-to-end latency and jitter; packet loss; and any IP-layer fragmentation or application-level segmentation resulting from the additional security overhead.

14.4.2 Mapping to the reference scenarios

The UC8 system model is instantiated through two reference scenarios that use the same UAV mission, MAVLink data, UDP transport and 5G communication infrastructure. The distinction between the scenarios is limited to the backend used to provide the required PiQASO cryptographic operations.

In **Reference Scenario 1**, the local backend is selected. The GCS and UAV invoke the PiQASO PQC SDK locally. On the UAV, long-term keys and security-sensitive operations are protected through the QR-TCB. The ZCU104 is additionally used to demonstrate a reduced-scope acceleration configuration for selected cryptographic operations that are available and integration-ready at the time of the UC8 experiments.

In **Reference Scenario 2**, the service backend is selected. The same GCS and UAV application software invokes the required cryptographic functionality through a PQCaaS client, which communicates with the PiQASO PQCaaS/PQ-DES backend. The mission application, PX4 integration, MAVLink encoder and decoder, UDP packetisation and flight-safety logic remain unchanged.

Table 14.2: UC8 mapping to the reference scenarios

System element	As-is baseline	Reference Scenario 1	Reference Scenario 2
UAV application protocol	MAVLink	Same MAVLink frames	Same MAVLink frames
Transport	UDP over 5G/IP	UDP over 5G/IP	UDP over 5G/IP
PiQASO backend	Disabled or absent	Local PQC SDK and QR-TCB	Remote PQCaaS/PQ-DES
Hardware acceleration	None	Both accelerated and non-accelerated	Not attributed to the UAV; possibly provided by the service
Dependency on external cryptographic server	None	No	Yes
Connectivity assumption	Normal C2 baseline	Normal C2 link	Stable PQCaaS service link assumed
Primary comparison	Baseline packet and processing characteristics	Local compute and reduced acceleration overhead	Remote-call and service-dependency overhead

14.4.3 Assumptions

A1 – Common mission and traffic baseline. All configurations shall execute the same repeatable waypoint mission using the same MAVLink message set, message ordering, telemetry rates and mis-

sion parameters. Differences observed between the configurations shall therefore be attributable to the PiQASO protection mechanisms and their execution modality rather than to changes in the UAV mission.

A2 – MAVLink encapsulation over UDP and 5G. To preserve industrial relevance and compatibility with existing UAV applications, the MAVLink protocol shall not be replaced or modified. The original encoded MAVLink frames shall be encapsulated into UDP datagrams and transported over a 5G/IP link. The experiments shall assess how the introduction of PiQASO protection affects MAVLink encoding and decoding, packet-processing time, packet-size expansion, effective throughput and IP-layer fragmentation or application-level segmentation. The packetisation policy and effective path MTU shall be recorded, because fragmentation depends on both the cryptographic overhead and the number of MAVLink frames carried in each datagram.

A3 – Use of both PiQASO modalities. UC8 shall integrate and evaluate both the local PQC SDK modality and the remote PQCaaS modality. The same mission application shall be used in both cases, and the active cryptographic backend shall be selected through configuration rather than through changes to the MAVLink, PX4 or mission-control code.

A4 – Reduced-scope hardware acceleration. UC8 shall test a toned-down and representative acceleration configuration on the ZCU104. The UC8 demonstrator does not claim to integrate or validate the complete PiQASO hardware-acceleration architecture. Only selected PQC operations or primitives that are available and integration-ready at the time of the demonstrator shall be accelerated. The exact accelerated operations, hardware configuration and limitations shall be reported together with a software-only PQC SDK baseline.

A5 – Stable PQCaaS service link. For the nominal performance evaluation of Reference Scenario 2, the communication path between the UAV/GCS clients and the PQCaaS server is assumed to be stable, continuously reachable and sufficiently provisioned for the required remote cryptographic operations. This assumption is introduced so that the overhead of the PQCaaS modality can be evaluated without conflating it with uncontrolled backhaul failures.

14.4.4 Scope of ABE

Attribute-Based Encryption remains in scope as an application-layer mechanism for selective disclosure of UAV-generated data in a multi-ground-station environment. The UAV, or the trusted PQCaaS service acting on its behalf, encrypts a selected data object under a policy expressed over recipient attributes. Candidate attributes may include the recipient organisation, operational role, mission identifier, security clearance and authorised data category.

The resulting encrypted object may be delivered to several ground or base stations, but only a station possessing valid attributes that satisfy the encryption policy shall be able to decrypt it. For example, general flight telemetry may be available to all authorised operators, while high-resolution imagery may be restricted to stations having both the relevant mission identifier and a payload-operator or incident-commander role.

ABE supplements rather than replaces PQ-TLS/KEM-TLS and AEAD. The latter mechanisms protect the C2 channel and authenticate the communicating endpoints, while ABE controls which authorised recipients may recover particular application data objects.

14.4.5 Common cryptographic backend interface

To prevent backend-specific cryptographic logic from being embedded in the UAV flight application or the GCS mission software, UC8 shall introduce a common cryptographic service interface between the

mission/session-management layer and the PiQASO cryptographic components. The interface shall expose provider-independent operations for capability discovery, credential provisioning, session establishment, re-keying, key encapsulation, signature generation and verification, acquisition of an authenticated session context, status reporting and error handling.

Two backend adapters shall implement this interface. The **local adapter** shall map the interface operations to the PiQASO PQC SDK and, on the UAV, to the QR-TCB and the available reduced-scope acceleration path. The **remote adapter** shall map the same operations to the PQCaaS client and the PQCaaS/PQ-DES backend. Backend selection shall be performed through deployment or runtime configuration. The GCS application, MAVLink encoder and decoder, PX4 integration and flight-safety logic shall not contain different code paths for local and remote cryptography.

14.4.6 Cryptographic Execution Paths and End-to-End Protection

UC8 distinguishes between two related but separate concepts: **end-to-end protection**, which describes whether data remains protected between its originating and intended receiving applications, and **cryptographic execution location**, which identifies where the cryptographic operations are physically performed. The two UC8 reference scenarios protect the same MAVLink-over-UDP application flows but use different cryptographic execution paths and trust boundaries.

In **Reference Scenario 1, the local PQC SDK path**, the Ground Control Station and the CERBERUS UAV execute the required post-quantum operations locally through the PiQASO PQC SDK. ML-KEM key generation, encapsulation and decapsulation, and ML-DSA signature generation and verification are performed at the endpoints. On the UAV, private-key operations and long-term key material are protected by the QR-TCB according to the selected QR-TCB configuration. The resulting session keys are used locally by the GCS and UAV to apply and remove AEAD protection to the MAVLink-over-UDP traffic. No external cryptographic service participates in the session-establishment or data-protection path. Consequently, both the cryptographic execution and the protected C2 data path are endpoint-to-endpoint.

In **Reference Scenario 2, the PQCaaS path**, the UAV and GCS invoke the same abstract cryptographic interface, but the remote backend adapter forwards selected post-quantum operations to the PiQASO PQCaaS/PQ-DES service over an authenticated service channel. The service path therefore includes request serialisation, client-to-service transmission, remote cryptographic processing, response transmission and client-side processing. These elements shall be measured separately from the C2 data-plane processing.

For real-time C2 traffic in Reference Scenario 2, PQCaaS is used for the selected session-establishment, credential-management and public-key operations. Once an authenticated session context has been obtained, the symmetric AEAD protection and unprotection of MAVLink-over-UDP packets shall remain at the GCS and UAV endpoints. This prevents each real-time C2 packet from requiring a round trip through the remote PQCaaS server.

A separate trust model applies where PQCaaS directly performs application-level encryption, decryption or ABE processing for data objects. If plaintext data or sensitive decryption material is made available to PQ-DES, PQ-DES forms part of the trusted execution boundary. Such a flow shall be described as service-assisted policy-protected data delivery, rather than as strict endpoint-only end-to-end cryptographic execution. The UC8 integration report shall identify, for each tested operation, the execution location, the entities that can access plaintext and key material, and the associated trust assumption.

Table 14.3: UC8 cryptographic execution paths and end-to-end protection

Aspect	Reference Scenario 1: SDK path	Reference Scenario 2: PQCaaS path
--------	--------------------------------	-----------------------------------

Application interface	Common abstract cryptographic interface	Same common interface
ML-KEM and ML-DSA execution	Local at GCS and UAV	Selected operations executed remotely by PQCaaS/PQ-DES
UAV private-key protection	QR-TCB, according to the selected configuration	PQCaaS/PQ-DES trust model for delegated operations
C2 packet AEAD	Executed locally at GCS and UAV	Executed locally at GCS and UAV after service-assisted session establishment
External service dependency	None	Stable service link required to PQCaaS server
Plaintext C2 visibility	Endpoint applications only	Endpoint applications only. AEAD remains local
Service-executed ABE	Not assumed as part of the SDK path	Used for the selected multi-ground-station data-sharing workflow
Main latency components	Local SDK and QR-TCB execution	Serialisation, service-channel latency, remote processing and local endpoint processing

14.5 Challenges and Needs from PIQASO

This section consolidates UC8 security needs, threats, and vulnerabilities, then maps them to PIQASO-provided mitigations that are explicitly in scope for UC8.

14.5.1 Security needs during UAV operations

To ensure safe and reliable missions, UAV architectures must be designed to satisfy the classic triad of Confidentiality, Integrity, and Availability (CIA), augmented by modern requirements for privacy, non-repudiation, and quantum-resilience. [41, 70, 64, 38]

- **Secure Communication and Data Integrity:** All mission-critical data—including command uplink, telemetry downlink, and sensor payloads—must be protected by authenticated encryption. This ensures that an adversary cannot read sensitive information or inject unauthorized commands. Data integrity is paramount; any modification of mission waypoints or flight configurations must be detectable via digital signatures or message authentication codes (MACs). Secure boot mechanisms are required to ensure that only untampered, verified firmware is executed by the onboard flight controller. [59]
- **Mutual Authentication and Access Control:** UAV systems require rigorous identity verification. Mutual authentication ensures that the GCS and UAV verify each other's credentials before establishing a link.[33] Access control must be granular and role-based, ensuring that different operators (e.g., flight pilot vs. payload operator) only have access to the functions necessary for their specific roles.
- **Availability and Resilience:** UAVs must maintain continuous, reliable operation even under adversarial conditions. This translates to needing resilience against jamming and denial-of-service on communication links. For example, a security requirement might be using spread-spectrum or frequency-hopping radios to mitigate simple jamming, [58, 26] or having an autonomous mode so the UAV can continue or safely abort its mission if the link is lost. High availability also means redundancy – backups for communication channels (satellite link as fallback to radio, etc.), backup navigation sources (e.g., using inertial navigation if GPS is jammed), and robust error handling.[3]

Real-time monitoring (health checks on link quality, intrusion detection systems) also supports availability by catching incidents early and triggering mitigations (like switching frequencies or invoking return-to-home on signal loss).[16]

- **Secure Data Storage:** Data security isn't only needed in transit but also at rest. UAVs require secure storage for any onboard recorded data, configuration files, and logs. This typically means encryption of the drone's storage media (SD cards, onboard memory) so that if the UAV is lost or stolen, the data remains confidential.[23] It also includes protecting data in ground control stations and cloud systems that store mission data – these should use strong encryption and access controls. In critical applications, end-to-end encryption might be necessary: data is encrypted by the drone's platform and only decrypted by authorized systems on the ground.[66] Additionally, secure storage implies regular key management (rotating encryption keys, using hardware security modules, etc.) to reduce the risk of key compromise. By ensuring stored data is secure, we prevent after-the-fact breaches or intelligence leaks if adversaries recover the hardware.
- **Data Integrity and Tamper Detection:** Beyond keeping data secret, UAV operations demand that data and commands remain unaltered from source to destination. The system must guarantee integrity – any corruption or tampering of flight commands, sensor readings, or software configurations should be detectable. This need extends to software integrity as well; secure boot mechanisms are required so the drone only runs trusted, untampered firmware.
- **Privacy and Regulatory Compliance:** In many UAV applications (especially civilian ones), protecting privacy and complying with regulations is a security need. Drones often capture video or sensor data that can include personal or sensitive information. Ensuring this data is handled according to privacy laws (e.g., GDPR) is crucial – which means access to the data must be strictly controlled and audited. Therefore, measures like anonymization of collected data, secure deletion after use, and strong audit trails for who accessed the data are necessary in certain missions (e.g. law enforcement or surveying). [37, 34]

14.5.2 Threat Landscape for UAV and Ground Station Communications

The communication link between the UAV and the Ground Control Station (GCS) represents a critical failure point that is targeted by a diverse array of active and passive threats.

- **Passive Interception and "Harvest Now, Decrypt Later":** Passive interception involves an adversary using high-gain antennas to capture the drone's downlink. While data may be encrypted today, adversaries often employ the "Harvest Now, Decrypt Later" (HNDL) strategy, where they store intercepted encrypted traffic to decrypt it once quantum computing capabilities become commercially available. This threat is particularly severe for long-term sensitive operations or defense-related intelligence gathering.[45, 14]
- **Insecure Communication Links:** Many UAVs use unencrypted or weakly protected command-and-control and telemetry links. [42, 71, 74] This makes it possible for attackers to intercept sensitive data or video feeds and even inject malicious commands if communications are not properly secured. For example, an unsecured radio or Wi-Fi link can be sniffed for intelligence or hijacked via man-in-the-middle (MITM) attacks. [35] Without robust encryption and authentication, adversaries could eavesdrop on live video streams or alter control signals in transit. This not only compromises mission confidentiality but also opens the door to spoofing and takeover attempts. [6]

In a MITM scenario, an attacker positions themselves in the communication path and impersonates the ground station to the UAV and vice versa. This allows them to alter or fabricate messages in real time. For instance, an attacker could intercept a command (“Return to base”) and modify it (“Land at these coordinates”) before relaying it to the drone. Without mutual authentication, the UAV would have no way to know the command came from a fake source. MITM attacks can be conducted by setting up a rogue relay station or by exploiting weaknesses in the radio protocol (e.g. forging session keys or exploiting a lack of encryption by establishing their transmitter as the strongest signal). The consequences range from subtle data manipulation (sending false telemetry back to the ground to mislead operators) to full control takeover if the attacker can inject navigation commands. This is one of the most severe threats because it effectively hands the drone over to the attacker’s control if successful.

- **Spoofing and Impersonation:** An attacker might not need to sit in the middle of an ongoing session if they can spoof one end of the link outright. Ground station spoofing is a known threat[28] - here, an adversary pretends to be a legitimate control station and sends commands to the UAV. This could happen if the UAV’s communication protocol doesn’t ensure the source is authentic, or if the attacker has stolen valid credentials/keys. Conversely, a drone could also be impersonated; an attacker could send fake status messages or data to the ground as if it came from the UAV (for example, spoofing a drone’s ID in a network to inject false fleet data). Impersonation threats also cover scenarios like session hijacking – if an attacker can guess or steal a session token or encryption key, they can take over an established session between the drone and ground. The result of successful spoofing can be devastating: the ground operator might follow false data (e.g. seeing a “all clear” status when the drone is actually in distress), or the UAV might obediently execute an attacker’s malicious commands thinking they’re authorized.
- **Denial of Service (Jamming):** An adversary might not want to steal data or take control – simply knocking the UAV offline can be enough to ruin a mission. By jamming the radio frequencies [40, 77, 72] used for UAV communications (commonly in the 2.4 GHz, 5.8 GHz, or specialized bands), an attacker can overwhelm the signal with noise. The UAV then either loses contact with the ground station or experiences a flood of errors. Many drones, when command link is lost, will attempt failsafe procedures (hovering, returning home, or landing). An attacker can exploit this by jamming at a critical moment (e.g. as a drone flies over a restricted area, causing it to fail or turn back). GPS jamming is another variant - by drowning out GPS signals, the drone may lose navigation capability or switch to a degraded mode. Jamming doesn’t require sophisticated equipment in some cases; even a crude radio frequency jammer can cause loss-of-link when placed sufficiently close or powered strongly. Beyond intentional jamming, flooding attacks at the network level (sending a barrage of bogus data or connection requests) can also clog the communication channel or ground station resources. The end goal is the same: disrupt the availability of the UAV’s control link, effectively blinding or stranding the drone.
- **Command/Telemetry Injection:** This threat involves inserting unauthorized messages into the communication stream without necessarily hijacking the entire session. If communications lack integrity checks, an attacker can inject a single malicious command (e.g., a navigation waypoint redirecting the drone) or inject false telemetry data. Command injection can exploit any unsecured channel: for instance, if there’s a secondary control interface or an open debug port, an attacker might send a carefully crafted command sequence that the UAV accepts. Even on encrypted links, poor protocol design (like not binding commands to a sequence number or time) can allow replay attacks [32, 10] - an attacker records a legitimate command and replays it later, causing the drone to repeat an action unexpectedly. A notorious example was the replay of “last known good” GPS signals in spoofing attacks, making a drone think it should continue a previous path. Without robust

sequence enforcement and authentication, these injection and replay attacks can manipulate drone behavior unpredictably.

- **Hijacking via Malware or Backdoor Communications:** An often overlooked threat is when the ground station or UAV is already compromised by malware [75] that opens a backdoor channel for an attacker. In such a scenario, the official communication link might be secure, but the attacker can issue commands through the malware. For example, if the ground control software runs on a laptop that got infected (perhaps via phishing or a malicious USB), that malware could intercept and alter commands before they even hit the radio link. Alternatively, the UAV's onboard computer could have a Trojan that waits for a special radio signal or network packet to wake up and then diverts control to the attacker. This is a "communications" threat in the sense that the attacker establishes an unauthorized parallel communication path (using an implant or malicious code). It underscores the need for end-to-end security: not just encrypting the link, but also ensuring the integrity of the systems that generate and consume the communication.

14.5.3 Vulnerabilities across the UAV mission lifecycle

The security posture of a UAV mission is not a static state but a dynamic equilibrium that must be maintained across a complex lifecycle involving pre-flight provisioning, in-flight command and control, and post-flight data retrieval. Analysis of contemporary UAV systems reveals a multi-layered vulnerability landscape spanning software, hardware, and physical domains.[43, 27, 62]

- **Software and Firmware Vulnerabilities:** UAV flight controllers and mission computers typically utilize ARM-based Linux or real-time operating systems (RTOS) that are susceptible to a range of classical cyber-attacks.[12, 4] A primary concern is the presence of buffer overflows, insecure Application Programming Interfaces (APIs), or undocumented backdoors within the onboard software stack. The "Maldrone" malware represents a paradigm shift in UAV-specific threats; it is the first backdoor specifically designed for ARM Linux UAV systems.[8] Maldrone operates by silently killing the legitimate autopilot processes and assuming direct control over the drone's device drivers and sensors.[1] Once established, the malware maintains persistence across system resets, allowing an attacker to intercept and modify data on the fly or drop the drone from the sky.

Furthermore, the firmware update process itself is often a point of failure [24] . If updates are delivered over unencrypted channels without cryptographic signature verification, an attacker can install a maliciously modified firmware image that grants them permanent control over the platform. This vulnerability is exacerbated by the use of third-party software libraries and open-source flight control stacks [36, 7] that may inherit unpatched vulnerabilities or contain hardware Trojans introduced during the supply chain process.

- **Hardware and Physical Attack Surfaces:** Physical access to a UAV, particularly one that has been recovered after a forced landing or crash, presents significant risks. Most commercial-off-the-shelf (COTS) drones lack robust tamper-resistance, meaning an adversary can access internal debug ports, retrieve unencrypted flight logs from SD cards, or extract cryptographic keys [69, 31] from non-volatile memory. Hardware Trojans and supply-chain manipulation are also critical concerns; maliciously altered chips can provide hidden backdoors that remain dormant until triggered by a specific mission event or RF sequence.
- **Communication Link and Network Vulnerabilities:** The wireless nature of UAV communication links makes them inherently vulnerable to interception and interference. Many UAVs still employ unencrypted or weakly protected command-and-control and telemetry links, allowing for passive

eavesdropping and intelligence gathering.[73, 11] In more complex scenarios involving UAV swarms or mesh networks, the system faces routing attacks such as "wormhole" and "Sybil" attacks. In a Sybil attack, a single malicious node illegitimately claims multiple identities, undermining the consensus and fault-tolerance mechanisms of the swarm. Wormhole attacks involve recording packets at one location and retransmitting them at another to disrupt the routing topology.

- **Network and Interface Vulnerabilities:** UAVs frequently use wireless interfaces (Wi-Fi, Bluetooth, telemetry radios) that can be entry points for attackers if not secured. For example, some drones create a local Wi-Fi network for control; if that network is on a default SSID/password or uses a weak protocol, any nearby attacker can connect. Additionally, open ports or services on the UAV or ground station can be discovered and attacked. Insufficient network security hardening (such as leaving debug interfaces active or not using firewalls) can let attackers gain footholds [2] . Drones operating in mesh or ad-hoc networks (as in swarms) face routing attacks like wormholes or Sybil attacks that can disrupt their coordination.
- **Navigation and Sensor Spoofing:** UAVs rely heavily on Positioning, Navigation, and Timing (PNT) data, primarily from Global Navigation Satellite Systems (GNSS) such as GPS. Because civilian GPS signals are unencrypted and low-power, they are trivial to jam or spoof.[30] Navigation spoofing involves broadcasting counterfeit GPS signals that lead the drone's flight controller to believe it is elsewhere, potentially causing it to veer off course or land in an attacker-controlled area.[63] Beyond GNSS, other sensors like Inertial Measurement Units (IMUs) and optical flow sensors can be targets of acoustic injection or optical interference, leading to total loss of flight stability.[29, 21]
- **Weak Authentication and Access Control:** Inadequate authentication mechanisms allow unauthorized parties to access the drone's controls or data. If a ground station or UAV does not strictly verify the identity of its communication partner, an attacker could impersonate a legitimate controller. Spoofing the ground station or sending forged commands can lead to an attacker secretly controlling the UAV. Strong, multi-factor authentication and role-based access are often lacking in legacy UAV systems, meaning an adversary with the right radio equipment or network access might bind to the UAV's link or network and issue commands. This is especially dangerous in military or critical infrastructure scenarios where drones carry out sensitive tasks.
- **Third-Party Component Risks:** UAVs often integrate third-party hardware modules or software libraries (for cameras, sensors, networking, etc.). If these components have known vulnerabilities, the drone inherits those weaknesses. For instance, a vulnerability in a 4G/LTE modem's firmware or an open-source flight control software could be exploited to compromise the entire UAV. Supply-chain attacks or trojaned components are a concern: a maliciously altered chip or module could provide a backdoor. Careful vetting and prompt patching of third-party components is often overlooked, leaving unpatched exploits available for attackers.
- **Insecure Data Storage and Privacy Gaps:** UAVs collect high-value data (video, sensor readings, surveillance information) which may be stored on the drone or transmitted to ground stations. If this data at rest is **not encrypted**, an attacker who breaches the drone or cloud storage could steal sensitive information. This is a critical vulnerability for missions involving reconnaissance or personal data (e.g. drones surveying private property). Moreover, storing data without proper access controls or segmentation could allow one compromised component (like a mobile app or ground control software) to leak all mission data. Privacy considerations are part of the security posture: UAVs might inadvertently expose personal data, so lack of data protection measures can violate regulations and create openings for social engineering or extortion.[47]

- **Lack of Monitoring and Incident Response:** Many UAV systems do not employ robust logging, intrusion detection, or real-time monitoring of their communications and operations. [17] This is a vulnerability in itself because it delays detection of breaches. If an attacker is manipulating a drone or quietly eavesdropping, the operators might not realize until after damage is done. Insufficient logging means forensic analysis post-incident is difficult – one cannot easily tell what data was accessed or how the drone was compromised. An absence of alerting (for example, no alarm on lost GPS lock or on an authentication failure) can turn a small breach into a full mission failure. Essentially, without monitoring, small security incidents go unnoticed and cascade into larger problems.

14.5.4 How the PiQASO Framework Addresses These Security Challenges

The PiQASO framework is designed to **future-proof UAV communications and operations** against the vulnerabilities and threats discussed. Acting as a comprehensive security solution, PiQASO introduces robust cryptographic protocols, hardware security, and services that directly mitigate the identified issues:

- **Quantum-Resistant Secure Communication:** PiQASO secures the UAV-ground station link by employing advanced post-quantum cryptographic (PQC) algorithms for encryption and key exchange. By establishing a quantum-resistant secure channel (for example, a PQC-enhanced TLS handshake), PiQASO ensures that all command-and-control and telemetry data is strongly encrypted and authenticated. This effectively thwarts eavesdropping and MITM attacks. The post-quantum handshake uses ML-KEM for quantum-resistant key establishment. Authentication of the communicating entities and relevant handshake artefacts is provided using ML-DSA digital signatures. The symmetric session keys established through the handshake are subsequently used with AEAD to protect the confidentiality and integrity of MAVLink command-and-control and telemetry traffic. With this level of protection, attackers are unable to read or alter the communications, eliminating the risk of information leakage or in-flight command injection via the link. Notably, PiQASO's cryptographic schemes are chosen to be resistant to quantum attacks, so the framework protects against "store-now, decrypt-later" threats as well. This means mission data remains confidential and authentic both today and decades into the future.
- **Strong Mutual Authentication and Authorization:** Before take-off, the GCS and UAV establish a mutually authenticated post-quantum session using ML-KEM-based key establishment and ML-DSA-based authentication according to the selected PQ-TLS or KEM-TLS profile. Only an authenticated and appropriately authorised GCS may establish the mission-control session. Command authenticity and integrity are subsequently enforced by the cryptographic context of the established session and the AEAD data-plane protection. ABE is not used to authenticate individual C2 commands.
- **Attribute-Based Protection of Data Delivered to Multiple Ground Stations:** ABE is used for selected application data that may be distributed to more than one ground or base station with different access rights. The data source specifies an access policy, and only stations possessing attributes that satisfy that policy can decrypt the protected object. This enables a single UAV mission to distinguish, for example, between a flight-control station, a payload-analysis station and an external monitoring station without providing every station with access to all collected data.
- **Data Integrity and Tamper-Proofing:** PiQASO addresses integrity through the combined use of encryption and digital signatures for both communications and stored data. Every piece of mission-critical data – be it a flight command, a telemetry report, or a stored image – can be cryptographically signed and/or MACed under PiQASO's scheme. This means that any alteration to data in

transit or at rest will be detected because the cryptographic verification will fail. For instance, if an attacker attempted a MITM to change a waypoint coordinate, the UAV's PiQASO verification layer would recognize the signature mismatch and reject the command. Similarly, PiQASO's data storage services (PQ-DS) encrypt and sign data chunks individually, so if an adversary tries to tamper with mission logs or sensor data stored in the cloud or on the device, those changes are evident and the data can be discarded as untrusted. By ensuring integrity, PiQASO mitigates injection and replay attacks – replayed commands won't carry the correct fresh signature or token and thus will be identified as duplicates or fakes. In essence, PiQASO creates a trusted computing environment where the drone will execute commands only if they have the right cryptographic credentials, thereby preventing unauthorized manipulation of the drone's behavior.

- **Secure Data Storage and Transmission (End-to-End Encryption):** The framework provides solutions for protecting data not just in transit but also at rest. Through its PQ-DS and cryptographic SDK, PiQASO enables quantum-safe encryption of stored data on the UAV and in ground systems. For example, when the UAV captures imagery or sensor readings, PiQASO can encrypt that data with a post-quantum cipher immediately. Only authorized parties with the correct PQ keys can decrypt it, which secures the data even if the drone is captured or a cloud server is compromised. This directly addresses vulnerabilities like insecure data storage and potential breaches – even if attackers get the files, they cannot decrypt the content. Moreover, PiQASO's approach uses granular key management: each data payload can be encrypted with a unique key (wrapped by a Kyber public key), and access policies determine who can unwrap it. This means that, for instance, video data could be encrypted under a key that only the mission commander's system can decrypt, ensuring privacy and compliance. Secure data exchange protocols in PiQASO cover the entire route from UAV sensors to ground user interface, maintaining encryption the whole way and thus shielding sensitive information throughout the mission lifecycle.
- **Protection Against Quantum Threats:** A cornerstone of PiQASO is that all its security measures are future-proof. By implementing NIST-recommended PQC algorithms (such as lattice-based schemes) for encryption, key exchange, and digital signatures, PiQASO removes the vulnerability posed by quantum computing advancements. Traditional RSA/ECC-based security could be broken in the future, but PiQASO's use of alternatives like Kyber, Dilithium, or Hash-based signatures ensures that even a quantum-capable adversary cannot compromise the UAV's communications or stored data. This future-proofing is critical in industries like aerospace and defense (the domain of Use Case 8) where mission data may remain sensitive for a long time. By integrating PQC at the protocol level (e.g. PQ-TLS and KEM-TLS), PiQASO guarantees that the confidentiality and integrity of UAV operations hold strong, even as new computational threats emerge. This essentially neutralizes the "record now, decrypt later" strategy – anything recorded from a PiQASO-protected drone will still be indecipherable later.
- **Resilience and Performance in Adversarial Environments:** PiQASO not only secures communications but also ensures the solutions are practical for real-world UAV conditions. The framework includes a Trusted Computing Base (TCB) component – in Use Case 8, this is realized as a hardware module (the ZCU104 FPGA board) on the UAV that accelerates cryptographic operations. By offloading complex PQC computations to dedicated hardware, PiQASO maintains low latency and high throughput on the secure link. This means the encryption/authentication overhead does not bog down the UAV's control responsiveness – an important factor in countering jamming or DoS. With efficient crypto, the channel can use spread-spectrum or rapid re-keying without undue delay, helping to mitigate jamming by quickly switching frequencies or sessions if an attack is detected. Furthermore, PiQASO's comprehensive testing in resource-constrained environments ensures reliability: the cryptographic schemes are benchmark-tested under various conditions (CPU load, link quality, etc.) to ensure they hold up when the drone is under stress. The framework's layered secu-

ity (combining hardware TCB, software SDK, and cloud services) also provides redundancy. For example, if the UAV loses connectivity to the cloud service, it still has the on-board PQC capability to continue secure operations. By validating its PQC modalities in a real UAV testbed, PiQASO demonstrates operational resilience – security measures remain effective without disrupting the mission even under adverse conditions.

- **Addressing Multiple Access and Swarm Scenarios:** PiQASO's design inherently supports scenarios with multiple ground stations or UAV swarms, which are often more complex security-wise. Through its PQC-as-a-Service model, the framework can act as a central trust anchor for many nodes. For instance, if multiple ground control stations need to communicate with one drone (or multiple drones), PiQASO's cloud-based key management service (PQ-DS) can distribute quantum-safe keys and certificates to all parties, ensuring they are part of a trusted group. This is particularly useful in swarm deployments where drones also communicate drone-to-drone: PiQASO can issue group keys or facilitate secure channels between UAVs for coordination. The framework's emphasis on policy-based encryption means that even in a multi-UAV network, each data or command can be tagged with which UAVs or stations are permitted to decrypt/handle it. In effect, PiQASO creates a secure communications mesh for swarms, mitigating threats like impersonation or data spoofing among swarm members. Additionally, having a unified PQC service allows consistent security updates and posture across all units – for example, if a new quantum-resistant algorithm emerges, it can be rolled out through the service to all drones and stations simultaneously. This centralized yet flexible approach means that even complex scenarios (multiple operators, hand-offs between control stations, large drone fleets) can be secured with the same level of confidence as a one-to-one link. PiQASO thereby solves the coordination security problem by extending its robust encryption and authentication across the entire operational network.

14.6 “To-Be” Reference Scenario #1: On-device PQC via PQC SDK and PQC-TCB/QR-TCB

The first “to-be” configuration scenario for Use Case 8 demonstrates a fully localized implementation of post-quantum secure communications, where both the ground control station (GCS) and the CERBERUS UAV execute cryptographic operations via the PiQASO PQC SDK. The airborne platform is further equipped with a dedicated PQ-Trusted Computing Base (PQC-TCB), deployed with QR-TCB acceleration through the AMD Zynq UltraScale+ MPSoC ZCU104 board. This on-device modality emphasizes autonomy, reduced dependency on backhaul networks, and real-time cryptographic responsiveness within resource-constrained airborne systems.

In this scenario, the mission involves a CERBERUS UAV executing a repeatable waypoint flight while maintaining quantum-resistant communications with one or more PiQASO-enabled ground stations. The cryptographic execution path is fully local. The GCS and UAV generate and maintain their respective ML-KEM and ML-DSA key material and execute the required key-establishment and authentication operations through the PiQASO PQC SDK. On the UAV, private-key operations are performed within the QR-TCB where required by the selected security configuration. Following successful session establishment, the two endpoints derive the symmetric session keys and locally apply AEAD protection to all MAVLink-over-UDP mission and telemetry traffic.

The PQ-TLS or KEM-TLS protocol forms the basis of secure channel establishment. Prior to arming, the ground station initiates the handshake sequence by negotiating PQ-capable modes via a ClientHello message. The UAV responds with the selected post-quantum handshake parameters through ServerHello. This is followed by the exchange of key encapsulation messages, which initiate and confirm the secure

derivation of symmetric session keys. Authentication proofs based on post-quantum digital signatures are then exchanged bi-directionally to finalize mutual identity validation. Once complete, the symmetric keys derived from this handshake are used to protect all mission and C2 traffic using authenticated encryption with associated data (AEAD) schemes.

Following the establishment of the secure channel, the second operational phase begins, encompassing the upload of mission parameters—including waypoints, geofencing constraints, and flight safety rules. This data is transmitted under AEAD protection using the keys negotiated during the handshake. Throughout the flight, the same secure tunnel is used for continuous telemetry and control traffic. The third operational phase involves in-flight security monitoring and contingency handling. Should the communication channel experience an integrity failure or authentication breakdown, deterministic safety responses—such as holding position, return-to-home, or emergency landing—are automatically triggered based on preconfigured mappings between cybersecurity anomalies and flight safety maneuvers.

Performance remains a critical consideration for this localized PQC implementation. To meet real-time requirements, PQ-heavy operations are limited to session initiation and periodic re-keying events. Continuous data-plane traffic is efficiently sustained via symmetric AEAD encryption, which minimizes latency overhead. The QR-TCB on the ZCU104 platform plays a pivotal role in this optimization by accelerating key encapsulation and signature verification, ensuring the UAV can maintain secure operations without compromising mission responsiveness.

Interoperability with existing avionics systems is also addressed in this configuration. The UAV's flight control logic, including the PX4 autopilot, must be fully aware of cryptographic channel states, enabling the system to respond appropriately to security events. Similarly, ground control interfaces must integrate secure session verification as a prerequisite for arming the UAV and initiating mission uploads. Although the exact specifications of certificate formats are not strictly defined in the current UC8 baseline, any such compatibility issues are treated as implementation-dependent and will be validated accordingly during integration.

Reference Scenario 1 establishes a concrete and autonomous security architecture for UAV operations, enabling quantum-resistant mission assurance with no dependency on external cryptographic infrastructure. It showcases how post-quantum technologies can be tightly integrated into airborne platforms using dedicated hardware acceleration and secure software stacks, while preserving interoperability, performance, and flight safety integrity.

14.7 “To-Be” Reference Scenario #2: PQCaaS-assisted PQC via PQ-DS for multi-ground and swarm scalability

Reference Scenario 2 evaluates the service-assisted PiQASO modality. The UAV and GCS use the same application-level cryptographic interface employed in Reference Scenario 1, but the interface is configured to use the PQCaaS adapter. Selected ML-KEM, ML-DSA, credential-management and session-establishment operations are therefore submitted to PQCaaS/PQ-DES over an authenticated service channel rather than executed through the local PQC SDK.

The real-time C2 data plane remains endpoint-based. After PQCaaS has provided the authenticated session context or cryptographic artefacts required by the agreed integration profile, the GCS and UAV derive or receive the applicable symmetric session material and locally apply AEAD protection to MAVLink-over-UDP traffic. Mission and telemetry packets are not required to transit through PQCaaS for per-packet encryption or decryption. The measured PQCaaS overhead therefore comprises the service request and response processing, the service-channel latency and the remote cryptographic execution time, while the packet-level AEAD overhead remains an endpoint measurement.

Reference Scenario 2 additionally evaluates ABE for the controlled delivery of selected UAV-generated data to multiple ground or base stations. For this workflow, data objects are encrypted under an access-control policy so that only stations whose certified attributes satisfy that policy can recover the data. The ABE workflow is separate from the continuously protected MAVLink C2 channel and is intended for selected data categories such as imagery, sensor products, mission reports or operational logs.

The architectural implications of this design extend well beyond cryptographic delegation. PQCaaS becomes the central point of control for credential provisioning, session lifecycle management, and key governance across multiple assets and domains. This centralization provides operational benefits particularly relevant to scalable mission scenarios: secure credential rotation, coordinated policy enforcement, and reliable logging for forensic and compliance purposes. In addition, it offers quantum-safe storage for data at rest. Mission telemetry, logs, and operational status data can be offloaded from the UAV to the remote server for secure retention and future audit, without imposing additional computational burden on the airborne platform.

The applicability of this model to multi-ground-station and swarming UAV scenarios is a core element of the UC8 design. Each UAV or ground station—regardless of geographic or operational domain—establishes a dedicated PQ-TLS/KEM-TLS session with PQCaaS. This capability enables seamless handover between control authorities, ensures consistent enforcement of secure session prerequisites, and supports real-time mission handoffs between operational domains. The governance structure embedded within PQCaaS also ensures that each node adheres to a centralized key lifecycle, certificate revocation, and data protection policy framework.

While this scenario is architecturally robust and operationally scalable, it introduces certain dependencies that must be addressed during evaluation. Most notably, mission execution becomes reliant on the reachability and availability of the PQCaaS over the backhaul network. Latency variance, intermittent service disruptions, or complete service outages could affect the timeliness of session establishment or the responsiveness of the security stack. UC8 explicitly considers these concerns by mandating comparative mission trials under both local (on-device) and service-based configurations. These trials are designed to quantify the impact of service dependence on mission readiness, session handshake latency, and fallback behavior in degraded connectivity conditions.

From a compatibility standpoint, the interoperability surface in the service modality is concentrated around the interface with PQ-DS and its integration with identity registration and policy enforcement mechanisms. All clients, whether UAVs or ground stations, must adhere to consistent identity bindings, ensuring the integrity of session authentication and traceability across operational domains. Moreover, downstream systems responsible for logging, auditing, or compliance enforcement must recognize PQ-DS as the authoritative source of cryptographic state and mission artifact records.

This reference scenario illustrates how PIQASO can extend post-quantum secure communications beyond individual UAV platforms to large-scale, distributed operational ecosystems. By enabling centralized cryptographic services, the PQCaaS model addresses not only the computational constraints of UAVs but also the governance, coordination, and operational continuity requirements of modern UAV deployments across civil, industrial, and defense sectors.

14.8 Reference scenario user stories and Metrics of success

This section provides UC8 user stories, confirmation criteria, workflow descriptions and success metrics aligned with the UC8 operational flow and PIQASO modalities.

14.8.1 User Story #1 (Reference Scenario #1): Secure pre-flight channel establishment as an arming gate

UC8.US1a: PQ-Secure Mutual Authentication as a Precondition for UAV Arming

As the UAV Operator and Ground Control Segment, I want to ensure the UAV refuses arming/take-off unless a mutually authenticated PQ-TLS/KEM-TLS secure tunnel is established with the authorized ground station so that the UAV cannot be armed or launched under the control of an unauthorized or unverified ground station.

UC8.US1b: Measurable and Logged Secure Channel Establishment for Audit Compliance

As the Mission Assurance Stakeholder, I want to ensure the secure channel establishment process is measurable and logged so that audit and post-mission compliance evidence is available to verify the security of the communication setup.

User Story Confirmation /workflow: Before any arming sequence is permitted and well in advance of take-off, the ground station and the UAV enter a mandatory security confirmation phase in which they attempt to establish a post-quantum-resilient secure channel using either PQ-TLS or KEM-TLS. This exchange is designed to produce a mutually authenticated tunnel, ensuring that both endpoints can prove their identities and that the command-and-control pathway is protected against impersonation and tampering. If, at any point, authentication fails or integrity guarantees cannot be achieved, the system responds with an explicit denial of arming authority. In parallel, the event is captured as an auditable record, ensuring that the denial is not only enforced in real time but also traceable for governance, compliance, and post-incident review.

Operationally, the workflow begins when the ground station initiates secure channel establishment across the C2 radio link. The implementation path then follows the selected modality. In a local modality, post-quantum operations are executed directly on the UAV's Zynq/QR-TCB and within the ground station SDK, enabling end-to-end tunnel formation without external dependencies. In a service modality, the UAV and ground station act as thin clients and interface with PQ-DS to obtain the required session material, after which the secure tunnel is instantiated using the provided cryptographic artifacts. Regardless of modality, the outcome of tunnel establishment is surfaced as a formal arming precondition within the ground control workflow, so that arming remains gated on verified secure connectivity rather than operator discretion or implicit assumptions about link trustworthiness.

14.8.2 User Story #2 (Reference Scenario #1): Secure mission upload and continuous C2 protection

UC8.US2a: PQ-Secure Upload of Mission Parameters to UAV

As the Ground Control Segment, I want to upload mission waypoints, geofencing parameters, and safety rules only through the established PQ-secure tunnel so that mission-critical configuration data cannot be injected or tampered with by unauthorized parties outside the secure channel.

UC8.US2b: AEAD-Protected In-Flight Telemetry and Command Traffic

As the UAV, I want to ensure all in-flight telemetry and command traffic is protected using AEAD (Authenticated Encryption with Associated Data) under symmetric keys derived from the post-quantum handshake so that any tampering with telemetry or command data is detected.

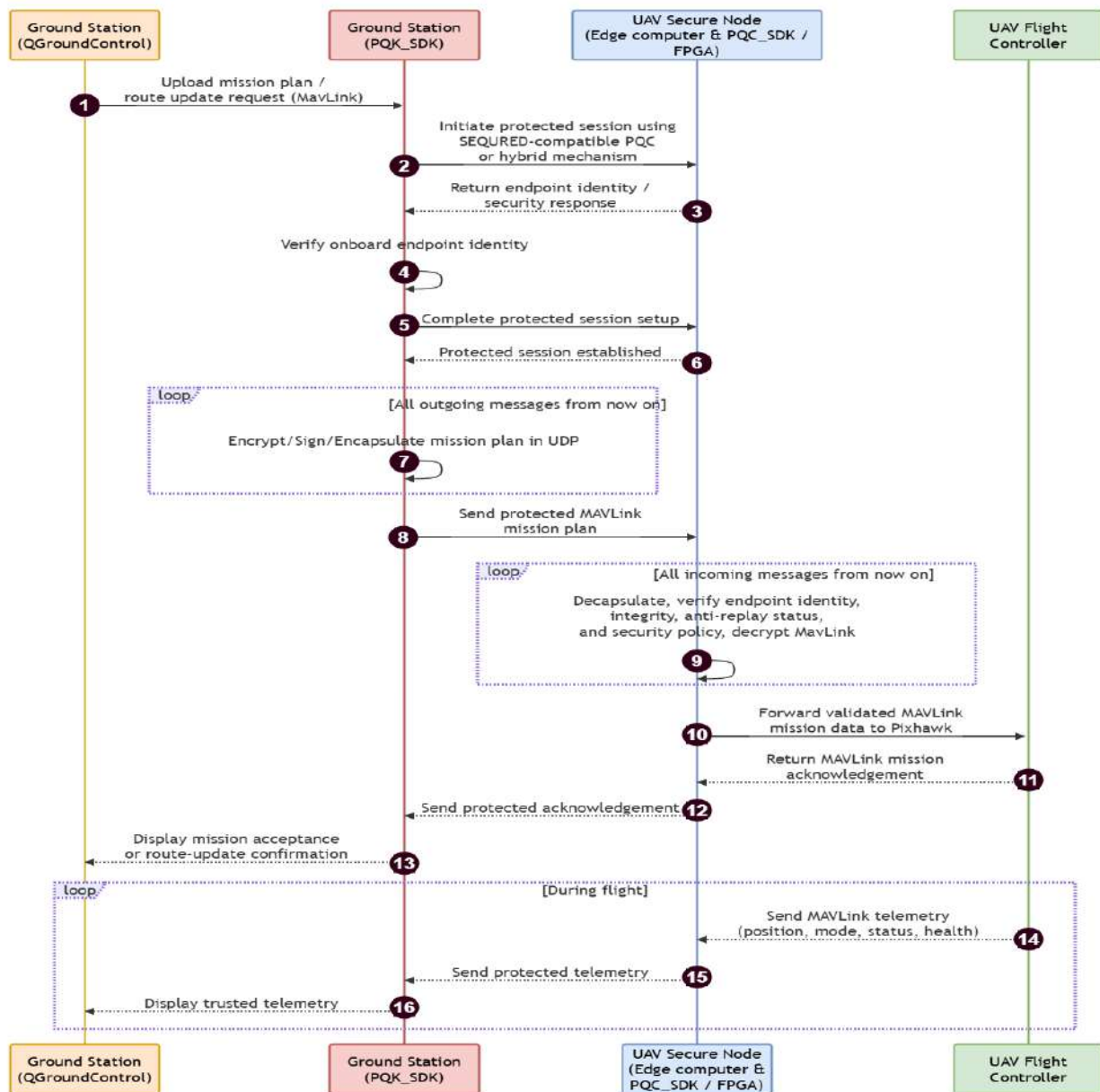


Figure 14.4: UC8 Sequence Diagram using onboard secure node

User Story Confirmation /workflow: Following successful establishment of the PQ-TLS/KEM-TLS secure tunnel, the mission upload proceeds and is accepted only within that protected session. The upload is cryptographically safeguarded end-to-end, ensuring that the payload is transmitted under strong confidentiality and integrity guarantees and that the receiving UAV can validate the mission data as authentic and untampered before it is applied.

Once the UAV transitions to in-flight operations, the command-and-control link remains encapsulated within the same secure tunnel, preserving continuity of protection throughout the operational lifecycle. The UAV enforces strict command acceptance criteria, requiring every command to verify correctly within the established cryptographic context; any command that cannot be authenticated and integrity-checked is treated as invalid and is not executed. AEAD protection is applied to C2 traffic to deliver both confidentiality and integrity at the transport layer, preventing unauthorized disclosure and providing robust detection of any manipulation attempts. If an integrity or authentication failure occurs at any point during flight, the system detects the anomaly immediately and transitions to controlled safety outcomes in accordance with predefined safety logic, thereby maintaining operational resilience and ensuring predictable, risk-managed behavior under adverse conditions.

14.8.3 User Story #3 (Reference Scenario #1): PQCaaS-enabled multi-ground and swarm governance

UC8.US3a: Centrally Governed Multi-Ground-Station UAV Interaction via PQCaaS

As a PiQASO-enabled Operator Group, I want to allow multiple ground stations to securely interact with one or more UAVs under centrally governed cryptographic policies via PQCaaS so that secure multi-station operations can be enforced consistently across the fleet without requiring any modification to aircraft hardware.

UC8.US3b: Reusable Secure Architecture for Cooperative and Swarm Missions

As a Fleet Operator, I want to use the architecture as a reusable blueprint for cooperative and swarm missions where multiple assets exchange mission data over untrusted networks so that secure multi-asset data exchange can be extended to future cooperative and swarm mission scenarios without redesigning the underlying security architecture.

User Story Confirmation / workflow: Following user story confirmation, each mission node—whether a ground station or an unmanned aerial vehicle—establishes its own post-quantum TLS relationship with the post-quantum directory service, using either PQ-TLS or KEM-TLS as dictated by the approved security baseline. This approach enables every node to operate within a centrally managed key-governance framework, ensuring consistent cryptographic control while still supporting multi-operator, multi-asset mission execution across heterogeneous platforms and organizational boundaries.

In the corresponding workflow, nodes first complete registration and authentication with PQCaaS in accordance with the applicable policy controls, including any required identity and compliance checks. Once admitted, the system brings secure sessions online so that mission communications and exchanges are conducted exclusively through PQ-secure tunnels, preserving confidentiality and integrity under post-quantum protections throughout the operational lifecycle. In parallel, governance-relevant logs and mission artefacts are retained in encrypted form to maintain evidentiary quality and access control, and they may be securely offloaded to PQ-DS to support audit readiness, compliance verification, and centralized oversight.

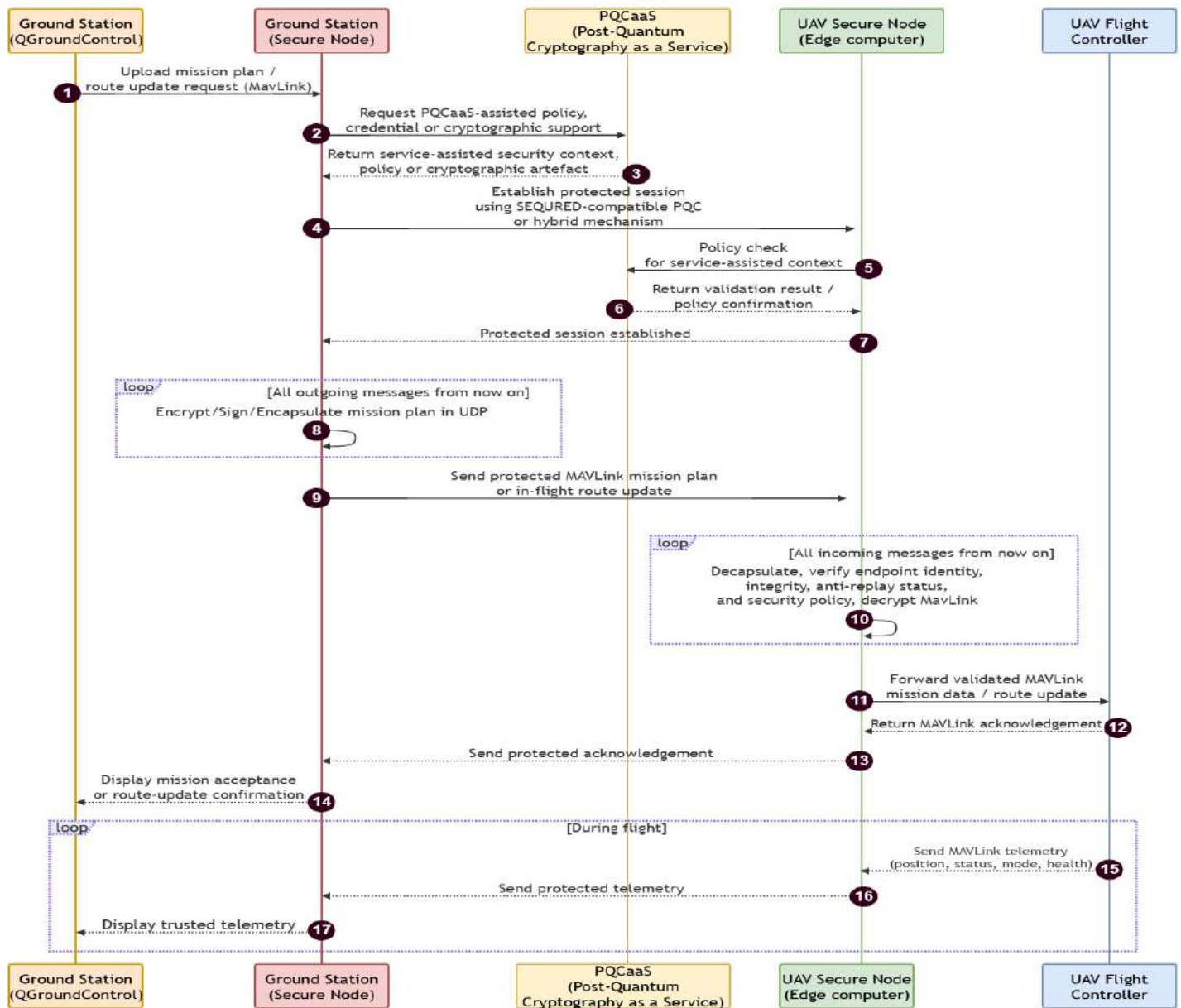


Figure 14.5: UC8 Sequence Diagram using PQaaS

14.8.4 Testing and validation approach (UC8 evaluation plan)

UC8_ready defines the mission as a repeatable waypoint mission executed under both modalities. The validation approach is therefore structured as follows:

- **Functional validation (both modalities):** i) Verify successful PQ-TLS/KEM-TLS establishment prior to arming/take-off, ii) Validate secure mission parameter upload within the tunnel, iii) Validate sustained AEAD protection for telemetry and command traffic iv) Validate deterministic safety responses to authentication/integrity failures.
- **Cryptographic execution-path and packet-performance benchmarking:** Measurements shall be reported separately for the local PQC SDK and remote PQCaaS paths. For the SDK path, measurements shall include local ML-KEM, ML-DSA, key-derivation and endpoint AEAD processing. Software-only execution and the reduced-scope hardware-acceleration configuration shall be reported separately. For the PQCaaS path, measurements shall separately include client request preparation, authenticated service-channel transmission, PQCaaS/PQ-DES processing, response transmission, client response handling and endpoint AEAD processing. For both paths, the evaluation shall also measure MAVLink encoding and decoding time, protected-packet size, bandwidth overhead, end-to-end latency, and UAV compute load.
- **Negative testing / adversarial validation:** i) Attempt session establishment using invalid credentials, altered handshake artefacts and replay-style misuse patterns in order to demonstrate correct rejection and fail-safe mapping; ii) alter or replay protected MAVLink-over-UDP packets to verify that they are rejected and not processed by the UAV; iii) verify that authenticated ground stations whose attributes do not satisfy an ABE policy cannot decrypt the corresponding protected data object; and iv) validate that security-relevant events are recorded in protected audit logs, including session failures, packet-integrity failures, ABE access denials and, where applicable, PQCaaS/PQ-DES interactions.

14.8.5 Deployment plan (integration and demonstration sequencing)

A pragmatic deployment plan consistent with UC8_ready is:

Phase 1 — Baseline mission integration (control reference):

- Establish repeatable waypoint mission and baseline telemetry characteristics.
- Confirm ground segment and UAV integration interfaces for mission upload and arming gates.

Phase 2 — Local modality integration:

- Integrate PQC SDK into ground segment and UAV; integrate PQC-TCB/QR-TCB concept on the UAV using ZCU104 compute/acceleration path.
- Demonstrate Stage 1/2/3 flows and capture KPI-US1/US2 metrics.

Phase 3 — PQCaaS modality integration:

- Configure ground and UAV thin-client behaviour; integrate PQ-DS channel and policy registration
- Execute identical missions and capture comparative metrics.

Phase 4 — Multi-ground / swarm-representative extension: Demonstrate multi-ground-station orchestration using PQCaaS as shared crypto layer; capture onboarding time and governance KPIs.

14.9 Metrics of Success

In Table 14.4, we provide the list of application-specific KPIs for this use case. Note that, as indicated by the given identifiers in the first column, each KPI is fulfilled by the corresponding user story, as described throughout this Chapter.

Table 14.4: Application-Specific Key Performance Indicators (KPIs) for the UAV Use Case

KPI	Description	Acceptance Criteria
[KPI-US1.1] Secure Channel Success Rate	Percentage of attempts resulting in a mutually authenticated tunnel under nominal operating conditions.	$p_{95} \leq 5\text{ s}$ for the selected UC2 profile.
[KPI-US1.2] Time-to-Secure Latency	Median and worst-case “ready-to-arm” latency, measured separately for SDK/QR-TCB vs. PQCaaS/PQ-DS configurations.	Protected $p_{95} \leq p_{95}^{\text{baseline}} + 100\%$ and absolute $p_{95} \leq 10\text{ s}$ for the agreed mission-size test file.
[KPI-US1.3] Negative Test Rejection Rate	Rate of correctly rejected attempts under negative testing scenarios (invalid credentials, altered handshake messages).	0 false accepts in the predefined negative test set; all rejected events must be visible to the operator or recorded in the validation log.
[KPI-US2.1] Command Latency Overhead	Added end-to-end command latency overhead compared to baseline (delta latency), measured under representative telemetry rates.	$\geq 90\%$ of telemetry accepted under nominal link conditions.
[KPI-US2.2] Bandwidth Overhead	Bandwidth overhead (bytes/second) attributable to secure encapsulation and associated metadata.	Added protection overhead $p_{95} \leq 1\text{ s}$ on 4G/5G, measured against the same-link baseline.
[KPI-US2.3] Fail-Safe Activation Rate	Correct fail-safe activation rate under induced integrity/authentication failures (hold / return-to-home / safe landing), with deterministic behaviour per configured safety policy.	After an outage of up to 10 s , the system either re-establishes a trusted session within 10 s or enters a visible fail-safe security state.
[KPI-US3.1] Ground Station Onboarding Time	Time to onboard an additional ground station, including credential provisioning, policy registration, and first successful secure session.	$T < 20\text{ s}$.
[KPI-US3.2] Concurrent Session Scalability	Ability to sustain $N = 3$ concurrent UAV sessions without governance failure or unacceptable latency (value of N to be agreed with integrator; aligns with assumptions A1/A2).	No governance failure or latency degradation beyond agreed thresholds across $N = 3$ simultaneous sessions.
[KPI-US3.3] Audit Log Evidence Quality	Completeness of audit log events, covering session establishment, mission upload, channel failure, fail-safe activation, and log offload actions.	$\geq 80\%$ overall event coverage and 100% coverage of induced channel-failure and fail-safe events.

Chapter 15

PiQASO UC9 – Sector: Transportation

15.1 Use case's Description, Goals and Objectives

This use case addresses railway monitoring through a drone-mounted or train-mounted inspection system operated by STROWL and consumed by the railway manager FCG. The inspection equipment includes an On-board Unit (OBU) equipped with high-resolution cameras that executes AI-based visual inspection during long-range Beyond Visual Line Of Sight (BVLOS) missions. OBU collects imagery over tracks and sidings, detect anomalous conditions (e.g., vegetation encroachment, obstacles, or structural track irregularities), and generate incident reports that include snapshots, location, and metadata. These reports are transmitted to a cloud backend that records them, exposes service APIs, and supports downstream workflows for ground monitoring teams composed of FCG personnel who retrieve, review, and act on the reported anomalies.

Within PiQASO, the goal is to make the end-to-end inspection pipeline resilient and future-proof by upgrading protection for mission configuration, command and control, anomaly reporting, and incident storage and retrieval. We will explore three scenarios: (i) ground control–OBU communications for mission setup and control, (ii) OBU–backend communication for reporting and ingesting anomalies and telemetry, and (iii) monitoring–backend communication for authenticated access and anomaly retrieval to support maintenance decisions.

Transportation infrastructure inspection faces security challenges that directly affect safety and service continuity. Communications may traverse heterogeneous and sometimes untrusted networks, and devices operate remotely with limited physical protection; in some operational settings an attacker may gain physical access to an OBU and attempt to extract secret keys, clone identities, or tamper with on-board software, making hardware-based key protection and tamper-resistance an important requirement [22]. Strong authentication is required to prevent impersonation of OBUs, operators, or backend services; confidentiality is needed to protect sensitive operational data and infrastructure information; and incident data must preserve integrity and provenance to prevent false alerts, tampering, or denial of maintenance actions [67]. Practical constraints also matter: OBUs may not be continuously connected to the internet, so once deployed, updating software, rotating credentials, or revoking compromised keys may be slow or infeasible without planned maintenance windows, increasing the value of cryptographic agility and long-lived protections [25]. In addition, because transportation infrastructure is critical infrastructure, mission artifacts and incident records may need to be retained for long periods (including years) for operational continuity, audits, investigations, and compliance, which amplifies the impact of confidentiality breaches and increases exposure to store-now-decrypt-later threats. Finally, these long-lived datasets are often shared across changing sets of authorized users; encrypting data at rest in a way that supports policy-based access and efficient user addition or revocation helps align confidentiality with operational needs over long time horizons while preserving auditability and controlled retrieval for operators and maintainers.

There is a clear need to migrate toward post-quantum cryptography because railway assets and the associated monitoring data have long lifecycles, and adversaries can capture traffic today and attempt to decrypt it in the future (store-now-decrypt-later). As public-key mechanisms that underpin authentication, key exchange, and secure channels may become vulnerable, maintaining confidentiality and authenticity for command/control and incident records requires a transition plan that anticipates cryptographic agility and long-term protection requirements. The migration itself introduces challenges that are particularly acute in this use case: drones and on-board units are resource-constrained and energy-sensitive, so implementations must be efficient while accommodating larger keys and higher computational costs; field devices may be intermittently connected, making timely patching, key rotation, and compromise recovery harder; and the risk of physical access to OBUs increases the need for hardware-backed key storage and acceleration to reduce exposure of long-term secrets. In addition, protecting long-lived incident repositories requires not only post-quantum secure channels (e.g., PQ-TLS) and post-quantum signatures and key encapsulation, but also post-quantum variants of cryptography-enforced policy mechanisms so that access to stored data can be governed and updated as roles change (e.g., attribute-based encryption and updatable public-key encryption).

15.2 PiQASO Functionalities

The Transportation Use Case will showcase the integration of specific PQC functionalities to provide secure control channels, robust authentication, hardware-based key protection, and cryptography-enforced access control. The following table details these functionalities, their specific application within the use case, and the planned reporting period for their demonstration.

Table 15.1: PiQASO Functionalities to be evaluated for Use Case 9 (Transportation)

PiQASO Functionality	Description and Purpose in the Use Case	RP1	RP2
PQ Signatures	Used by operator so sign mission plans.	✓	
PQ-TLS	Establishes a secure communication between various endpoints.	✓	
QR-TCB & HW Acceleration	Used for storing keys in drones.	✓	✓
ABE	Used for encrypting anomaly reports		✓

15.3 “As-is” Scenario

In the current (“As-is”) setup, operators prepare mission plans using a mission control application, sign them, and upload them to the OBUs over TLS-protected channels. During a mission, on-board AI models process camera imagery to detect anomalies; when an anomaly is identified, the OBU generates an incident report, signs it, and transmits it to the cloud backend over HTTPS. The backend stores the received incident records in its database (including associated metadata and evidence) and makes them available to downstream consumers. Ground maintenance personnel and FCG authenticated staff retrieve incidents from the cloud backend through a monitoring system. The monitoring system interacts with the backed using an HTTPS REST API. Then the monitoring systems uses maintainer-specific internal tools and processes to manage and act on the retrieved incidents. Figure 15.1 illustrates the architecture of the “as-is” scenario

Security in this “As-is” scenario relies on conventional TLS for securing communication channels, a PKI-based approach for endpoint authentication, ECDSA for digital signatures, and AES-based encryption for

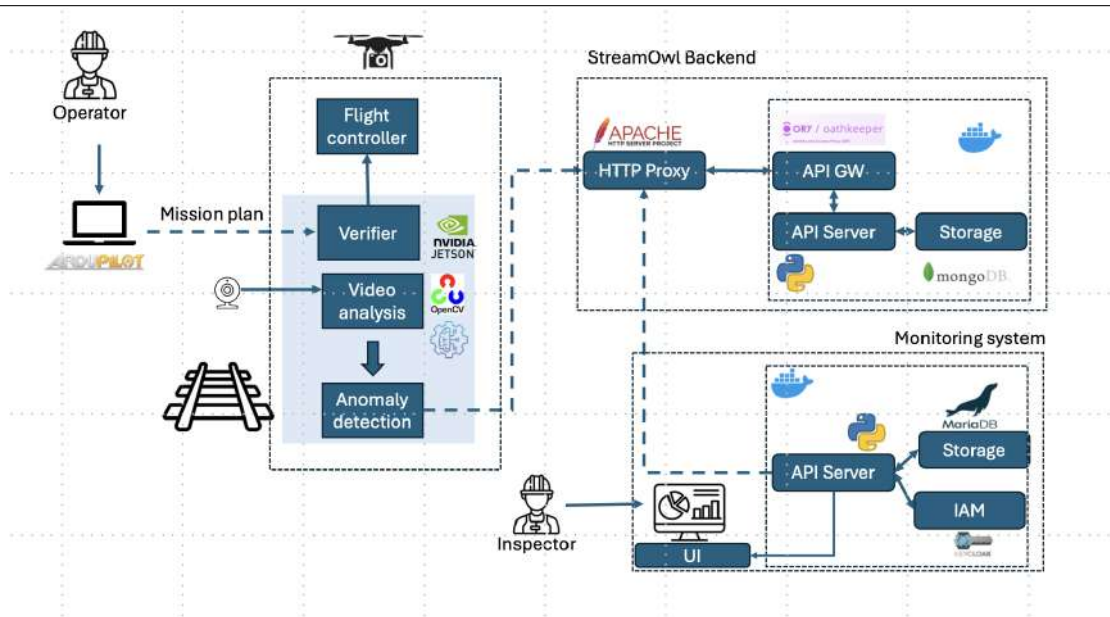


Figure 15.1: Architecture of the “as-is” scenario.

protecting data at rest. Maintenance personnel are typically authenticated at the application layer (e.g., username/password or API tokens) when accessing anomaly reports.

In more detail, STROWL operates a trusted key infrastructure that issues public/private key pairs used to identify ground control equipment operators and OBUs. Operators establish a TLS connection to the OBU over a controlled or private network, and upload the signed mission plan. The OBU validates the operator’s signature against a list of public keys and accepts the plan only if the signature is valid. For reporting, the OBU connects to the backend via HTTPS and submits signed anomaly reports. The backend verifies the OBU signature using the associated public key and then encrypts and stores the report. Backend’s HTTPS interface is protected using a certificate issued by Let’s Encrypt). Authorized users retrieve incident data after completing application-level authentication to their monitoring system. The monitoring system authenticated to the backend using an HTTP bearer token. In this baseline configuration, private keys are typically protected by the operating system and software keystores rather than dedicated hardware security modules.

15.4 Components and Roles Considered

The UC9 scenario involves the following actors, components, and systems:

- **OBUs:** The OBUs perform BVLOS inspection missions over railway assets using cameras and other sensors. They host an on-board computing stack for mission execution and local analytics, and they communicate with the backend over wireless links to transmit telemetry and signed anomaly reports together with supporting evidence and metadata. It executes AI models for anomaly detection. It prepares the data products that are sent to the backend (e.g., incident reports and evidence packages), applies cryptographic protections such as signing and secure communications, and maintains locally stored material such as mission configuration, device identifiers, and cryptographic key material.
- **Ground Control Station (GCS):** The GCS is used by operators to plan and configure missions (e.g., inspection paths, inspection targets, and schedules) and to deploy these mission packages to the OBU.

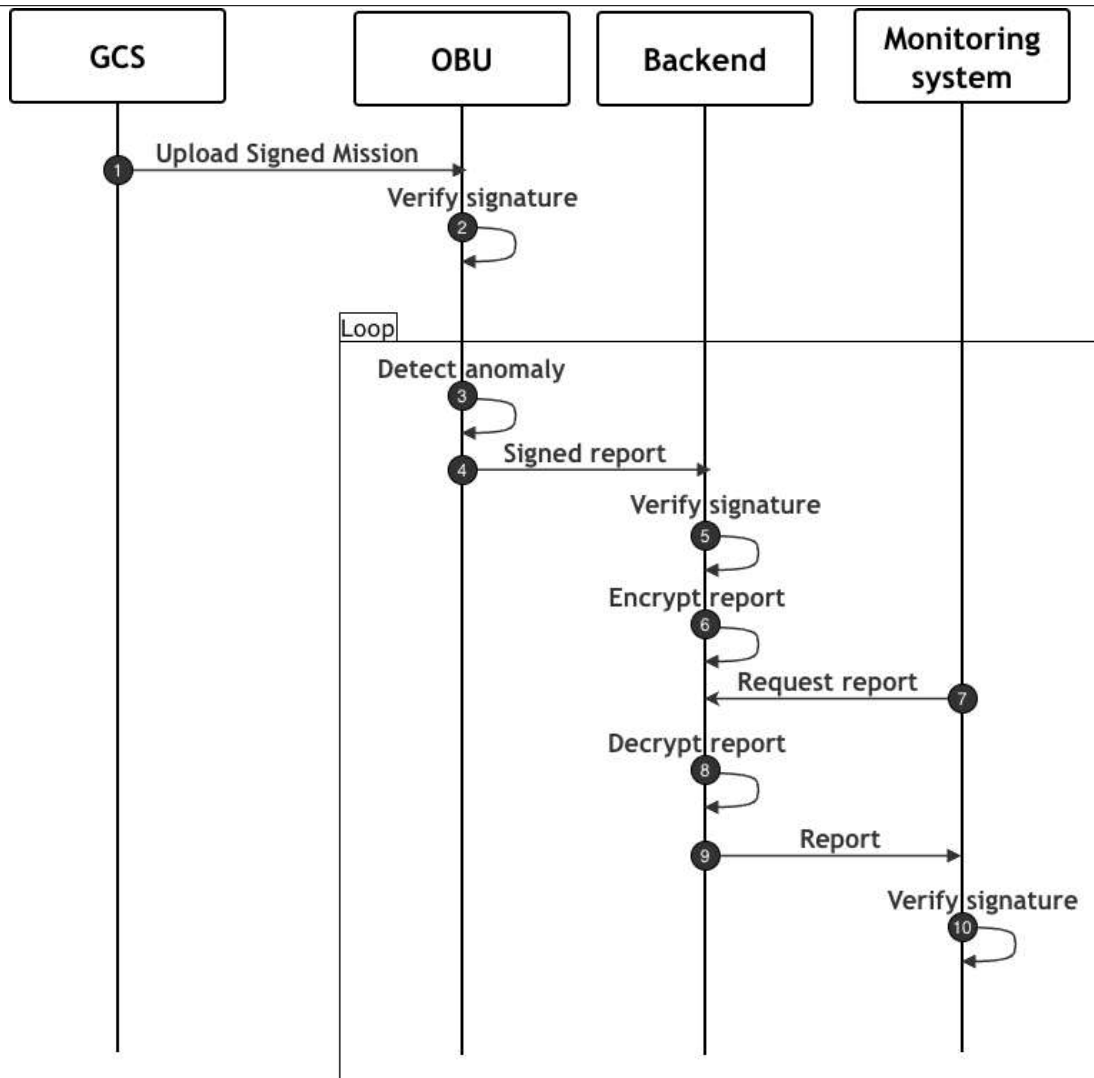


Figure 15.2: Sequence diagram of the “as-is” scenario.

- **STROWL Cloud Backend:** The backend receives anomaly reports from OBU, verifies their authenticity, and persists incident records and associated artifacts in databases and object storage. It exposes web dashboards and APIs for incident management, correlation, and visualization, and it integrates with notification and workflow mechanisms used by maintenance teams.
- **FGC Railway Manager Monitoring system:** FGC consumes incident information for operational decisions and maintenance planning, either via dashboards or API integrations into existing asset management and decision-support systems. It provides authenticated user access for relevant roles (e.g., maintenance engineers, dispatchers, and safety managers) and supports the processes required to retrieve, review, and act upon anomalies.

15.5 Challenges and Needs from PiQASO

The UC9 environment introduces technical and operational challenges that PiQASO must address to enable a realistic transition to post-quantum security across OBU, GCS, backend services, and monitoring systems.

A first challenge is the resource-constrained nature of OBUs. Cryptographic operations must be imple-

mented efficiently to avoid excessive CPU load, memory footprint, latency, and energy consumption while still providing strong security guarantees. This is especially relevant when introducing post-quantum algorithms, which may increase key sizes and computational overhead, and when secure channels and signing must be used routinely during mission deployment and anomaly reporting. In addition, because OBUs may be physically captured or inspected by adversaries, there is a need for hardware-backed keys (secure storage and, where possible, hardware-assisted cryptographic operations) to reduce the risk of secret-key extraction and identity cloning.

Secure mission configuration is another critical requirement. Mission packages contain sensitive operational details (e.g., routes, inspection targets, and constraints) and directly influence safety and compliance. The pipeline therefore needs strong authenticity, integrity, and confidentiality so that only authorized operators can deploy missions, OBUs can verify mission provenance, and attackers cannot tamper with or replay mission instructions.

The authenticity and integrity of anomaly reporting must also be strengthened. The backend needs high assurance that an anomaly report originates from a legitimate OBU and that the reported evidence has not been altered in transit. This requires robust signing and verification workflows and careful identity management.

Long-term secure storage and controlled sharing are essential because transportation infrastructure is critical infrastructure and monitoring artifacts may be retained for years. Protecting data at rest must account for long-lived confidentiality requirements and "harvest-now-decrypt-later" threats, while also supporting evolving access needs across multiple stakeholders. PiQASO should therefore support quantum-safe protection for data at rest and enable cryptography-enforced policy mechanisms (e.g., attribute-based encryption and updatable public-key encryption) that simplify adding, removing, and re-authorizing users over time without disruptive re-encryption workflows.

Finally, deployment constraints require interoperability and incremental migration. Legacy components and operational procedures will continue to rely on classical cryptography, and OBUs may be intermittently connected, limiting the ability to patch software or change cryptographic algorithms once deployed. PiQASO must facilitate transition through usable SDKs and services.

15.6 “To-Be” Reference Scenario #1: Quantum-Safe Mission Configuration

In the first reference scenario, PiQASO is integrated into the mission configuration and control pipeline so that every mission deployment to OBUs is protected by quantum-safe cryptography.

In the preparation and registration phase, each OBU is equipped with a QR-TCB PiQASO FPGA that is used to generate key material and offload post-quantum cryptographic operations. In parallel, the Ground Control Station (GCS) integrates the PiQASO SDK and provisions the corresponding credentials required to authenticate the operator and establish protected sessions with the drone.

When creating a mission package, the operator defines a new inspection mission (e.g., route, altitude profile, and expected inspection segments) and uploads it to the GCS. The GCS invokes the PiQASO SDK to sign the mission package using a post-quantum signing algorithm and establishes a PQ-TLS session with the drone.

For secure mission deployment, the signed mission package is transmitted to the on-board unit (OBU) via the GCS over the established PQ-TLS channel. The OBU uses its QR-TCB capability to verify the post-quantum signature and confirm that the package originates from an authorized mission planner. Only if all verifications succeed does the OBU accept and activate the mission.

In this To-be scenario, all critical mission configuration and control flows are protected with PQC-enabled authentication and encryption, significantly increasing resilience against both classical and quantum-enabled attacks.

15.7 “To-Be” Reference Scenario #2: Secure Data Storage and Retrieval in the Cloud

The second reference scenario focuses on the secure, quantum-safe storage and retrieval of monitoring data in the cloud backend, as well as controlled access by FCG monitoring system.

In this scenario, the OBU reports mission telemetry and detected anomalies to the STROWL backend over PQ-TLS. For each anomaly report, the OBU generates a post-quantum signature (using its QR-TCB capability) to bind the incident to a specific OBU identity and mission context. Upon receipt, the backend uses the PiQASO SDK to verify the post-quantum signature and enforce provenance checks before persisting the incident. Incident data at rest is encrypted using PQ-SDK, enabling long-lived confidentiality for archived monitoring artifacts.

Quantum-safe access by FCG is provided through a secure web portal and service APIs. When a monitoring system requests access to an incident, the backend retrieves the encrypted data from storage and delivers the content to the monitoring system over a post-quantum secure channel. The monitoring system then decrypts the data, verifies its signature using the PiQASO SDK to ensure end-to-end integrity and provenance.

This reference scenario provides **quantum-safe confidentiality of monitoring data**, ensuring that inspection images, videos, and anomaly reports remain confidential even in the presence of a future quantum computer, mitigating “harvest-now-decrypt-later” risks. It also strengthens **integrity and provenance guarantees** by binding every anomaly report to a specific OBU through post-quantum signatures, enabling FCG to detect tampering and prevent forgery. Finally, it supports a **smooth migration path** by encapsulating post-quantum capabilities behind PiQASO’s SDK and services, allowing existing cloud backends and client applications to adopt PQC with limited changes to business logic.

15.8 Reference scenario user stories and Metrics of success

15.8.1 User Story #1 (Reference Scenario #1): Quantum-Safe Mission Upload (Operator)

UC9.US1a: Authenticated and Tamper-Proof Mission Configuration Upload to OBU

As an Operator, I want to upload mission configuration to an OBU (On-Board Unit) such that the mission is accepted only if it originates from an authorized source so that the mission configuration cannot be modified or injected by an attacker.

Confirmation:

- The mission planner’s GCS uses the PQ-SDK to sign mission packages with PQC.
- The OBU successfully verifies the PQ signature and stores the mission.
- Any mission package that fails verification is rejected and logged.

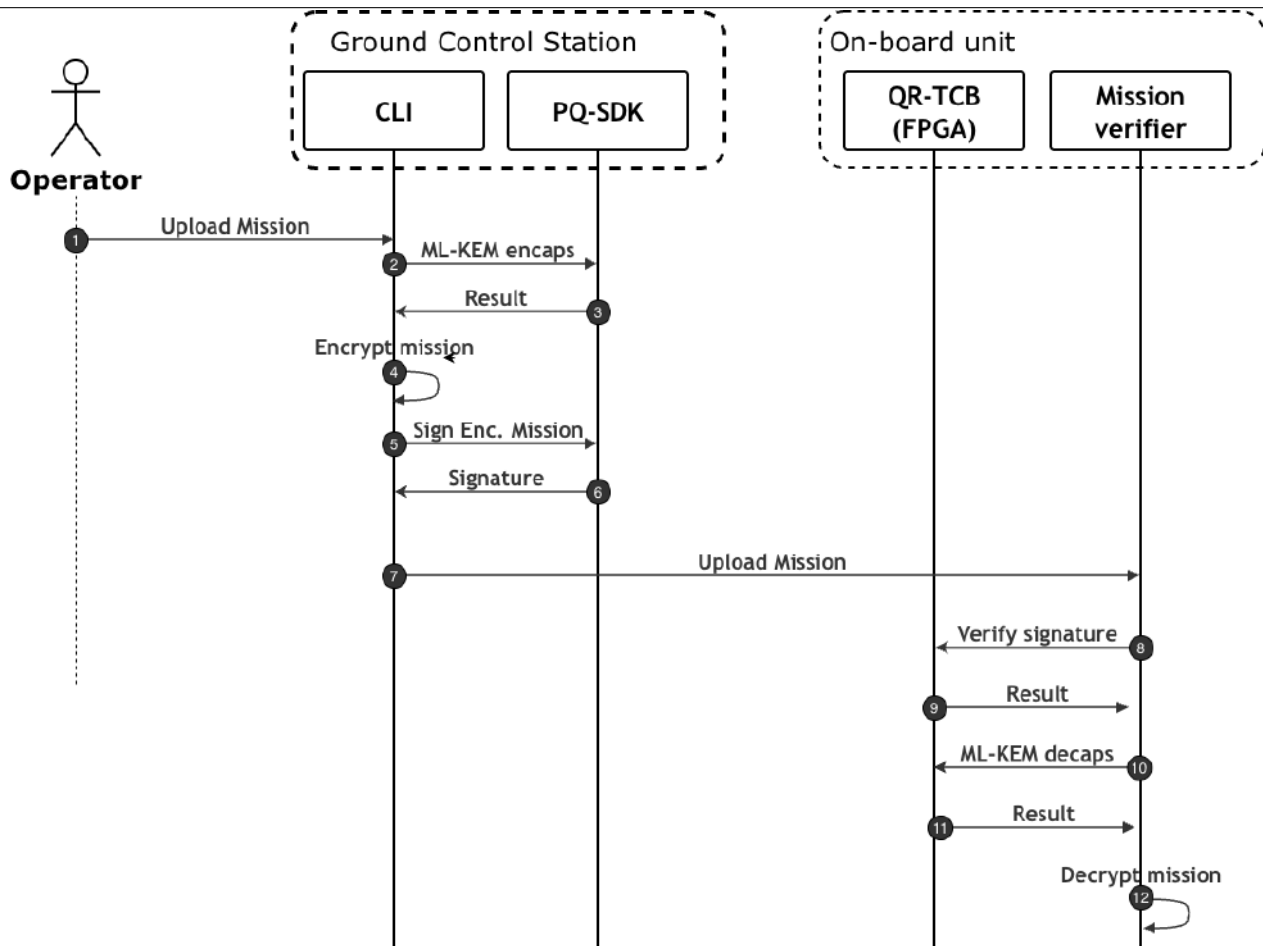


Figure 15.3: Sequence diagram of User Story #1.

User Story Workflow

- Operator uploads a mission package (route, parameters) to the GCS
- GCS requests from PQ-SDK an encapsulated key using ML-KEM and the public key of the OBU, it derives an AES-GCM key from the shared secret, and encrypts the mission.
- GCS signs the encrypted mission using the ML-DSA functionality of PQ-SDK and the operator's public key.
- GCS sends the encrypted and signed mission package to OBU using HTTP post
- OBU retrieves the public key of the operator through an internal data structure that includes public keys of approved operators and verifies the signature using QR-TCB
- OBU decapsulates the received ML-KEM ciphertext and learns the shared secret
- OBU derives the AES key and decrypts the received package

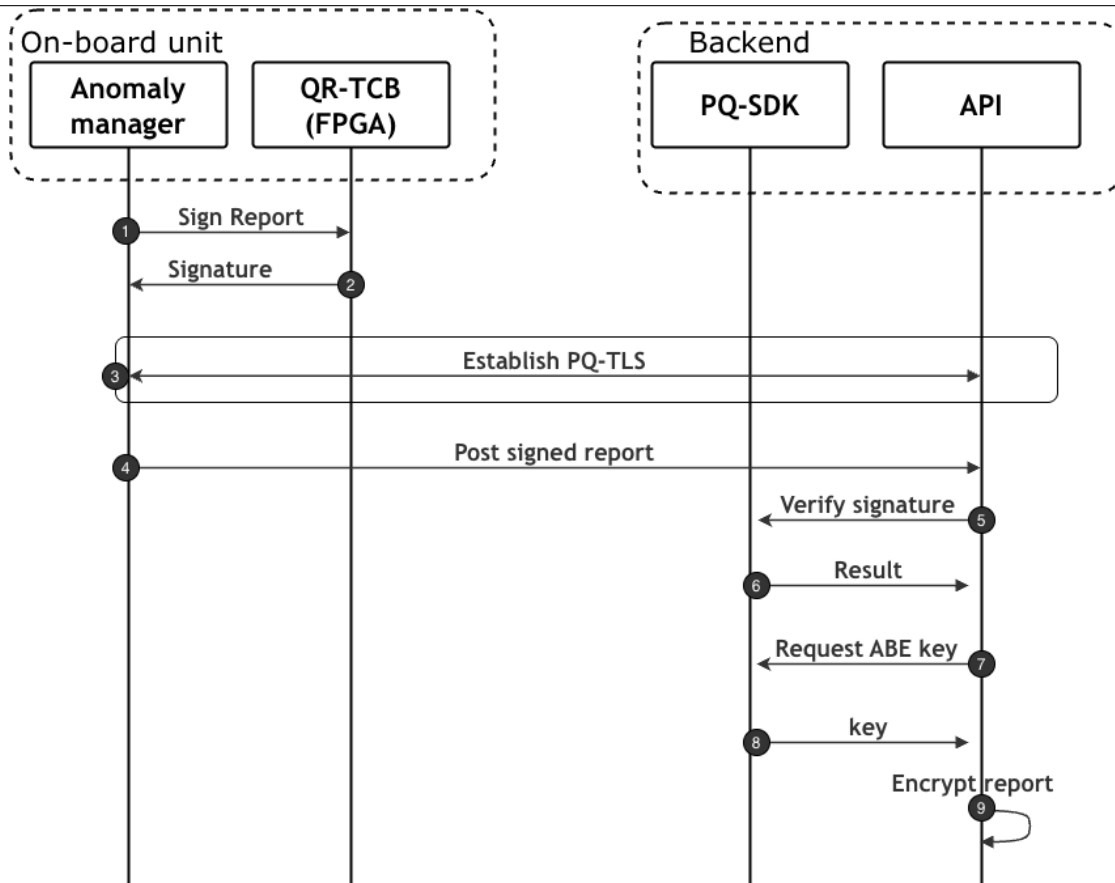


Figure 15.4: Sequence diagram of User Story #2.

15.8.2 User Story #2 (Reference Scenario #1): Authentic Anomaly Reporting (OBU)

UC9.US2a: Provable Origin and Tamper-Proof Anomaly Reporting from OBU

As an OBU, I want to send anomaly reports to the backend such that each report is provably generated by me so that the origin of the report is verifiable and its contents cannot be altered in transit.

Confirmation:

- Each anomaly report is PQ-signed before transmission using the QR-TCB.
- The backend verifies the signature using the registered OBU identity.
- Reports with invalid signatures or inconsistent identities are discarded and flagged as potential attacks.
- Valid are encrypted using PQ-SDK

User Story Workflow

- OBU detects anomaly via AI model.
- OBU prepares anomaly report (image, GPS, timestamp, mission ID).

- OBU signs the report using ML-DSA as provided by the QR-TCB and its own private key.
- Report is transmitted to backend over PQ-TLS established using either PQ-SDK or PQCaaS.
- Backend retrieves the public key of the OBU through an internal data structure that includes authorized public keys and verifies signature using PQ-SDK.
- Backend requests an ABE key from PQ-SDK associated with an appropriate policy
- Backend encrypts the signed report using ABE and stores it

15.8.3 User Story #3 (Reference Scenario #1): Secure Anomaly Retrieval (FCG Maintenance Engineer)

UC9.US3a: Confidential and Integrity-Protected Retrieval of Historical Anomaly Data

As an FCG Maintenance Engineer, I want to retrieve historical anomaly data (images, locations, timestamps) in a way that guarantees confidentiality and integrity so that sensitive anomaly data remains protected from disclosure and tampering, even against future quantum-capable attackers.

Confirmation:

- Access to anomaly data requires PQC-protected authentication and authorization.
- Retrieved data is decrypted by the monitoring system using PQ-SDK.
- Any attempt to access data without proper authorization is denied and logged.

User Story Workflow

- FCG engineer authenticates to web portal of the monitoring system and requests new anomalies reports.
- The monitoring queries the backend over PQ-TLS, established using PQ-SDK or PQCaaS, for incidents for a specific line/time window.
- Backend retrieves encrypted anomaly data
- Backend decrypts the report and transmits it over the established PQ-TLS channel.
- The monitoring system retrieves the public key of the OBU that generated the report, using an internal storage of authorized public keys, and verifies the signature using PQ-SDK or PQCaaS.

Metrics of Success (KPIs)

To assess the success of UC9, quantitative KPIs are defined across security, performance, reliability, and usability dimensions.

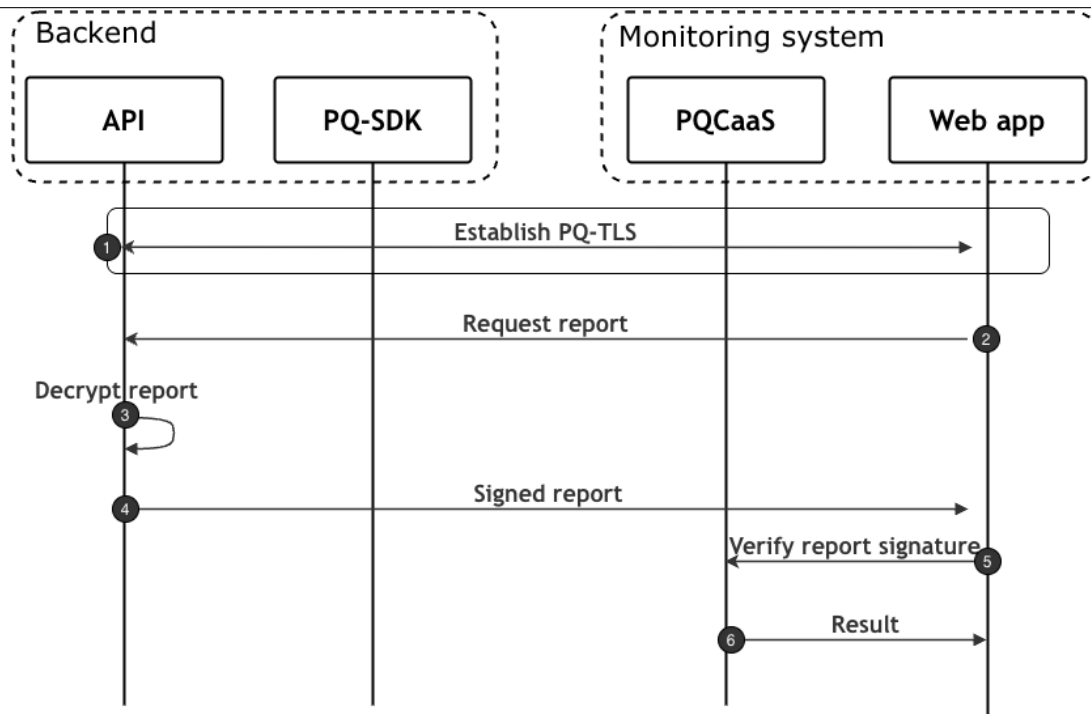


Figure 15.5: Sequence diagram of User Story #3.

Table 15.2: KPIs for evaluating UC9 success across security, performance, reliability, and usability. Concrete values for X, Y and SLAs will be determined during implementation and trials.

KPI	Description	Acceptance Criteria
Additional latency during mission upload	Mission upload requires PQC signature generation, PQ-TLS communication channel establishment, signed item transmission, and signature verification.	< 30% increase
Additional latency per submitted anomaly report	Anomaly report submission requires PQC signature generation, PQ-TLS communication channel establishment, signed item transmission, signature verification and encryption of the report using ML-KEM, or PQ-ABE.	< 50% increase
Additional latency per retrieved anomaly report	Anomaly report retrieval requires PQ-TLS communication channel establishment, encrypted item transmission, decryption of the report using ML-KEM or PQ-ABE, and signature verification.	< 40% increase
Additional bandwidth consumption	Increase in network bandwidth consumption caused by the transmission of PQC signatures and PQ-TLS handshake messages.	< 35% increase

Chapter 16

PiQASO PQC Artifacts Testing Plan per Use Case

Table 16.1: Functionalities for the Automotive use case

Functionality	Description	Status
QR-TCB	The Quantum-Resistant Trusted Computing Base (QR-TCB) of PiQASO is designed as the minimal trusted domain responsible for protecting long-term cryptographic assets and enforcing secure execution of Post-Quantum Cryptography (PQC). It is required in order to support the crypto capabilities detailed throughout this Table.	M18
PiQASO Software Library	In the Automotive use case, there is a need for establishing secure Vehicle-to-Infrastructure (V2I) communication, i.e., between the Vehicle On Board Unit (OBU) and the Roadside Unit (RSU). To this end, the PiQASO Software Library offers highly optimized implementations of the standardized ML-KEM and ML-DSA NIST-standardized schemes, which can be used in order to establish secure and authenticated communication channels between vehicles and infrastructure. In this case, both pure PQC and hybrid approaches are explored, where classical and post-quantum crypto primitives coexist to ensure backward compatibility during the transition period.	M18
PQC-as-a-Service (PQCaaS)	The PiQASO PQCaaS component is able to provide post-quantum crypto functionalities to devices which may not have the computational capabilities to do so on their own. As described in detail in D3.1, the PQCaaS component offers the capability for data encryption at the side of the PQ Data Encryption Server, so that such devices do not need to perform the encryption themselves. In the Automotive use case, the PQCaaS architecture is leveraged in order to meet the strict timing requirements of such applications (message latency typically required to be $\leq 100\text{ms}$) while providing quantum-resistant security.	M36
Updatable Public Key Encryption (UPKE)	Secure storage is considered as a requirement of the Automotive use case, in order to ensure the security of data stored on the infrastructure side. The PQCaaS component of PiQASO provides Forward Secrecy through the Updatable Public Key Encryption (UPKE) capability, which ensures that exposure of a secret key at a given point in time does not compromise the confidentiality of data protected in the past. This is a particularly pertinent need in cloud environments, where servers may be exposed to attacks and breaches may remain undetected for extended periods.	M36
Hardware Acceleration	The architecture of the Automotive use case leverages the hardware acceleration capabilities of PiQASO, as PQC algorithms are computationally more intensive than their classical counterparts. The identified architectures incorporate dedicated hardware acceleration through Field-Programmable Gate Arrays (FPGAs) to ensure that cryptographic operations meet the rigorous latency requirements of vehicular safety applications.	M36

Table 16.2: Functionalities for the Automation use case

Functionality	Description	Status
QR-TCB	The Quantum-Resistant Trusted Computing Base (QR-TCB) of PiQASO is designed as the minimal trusted domain responsible for protecting long-term cryptographic assets and enforcing secure execution of Post-Quantum Cryptography (PQC). It is required in order to support the crypto capabilities detailed throughout this Table.	M18
PiQASO Software Library	As previously mentioned, the PiQASO Software Library offers highly optimized implementations of the standardized ML-KEM and ML-DSA NIST-standardized schemes. In the context of the Automation use case, signature schemes (such as ML-DSA) are needed by the Fleet Manager (FMA) to sign task assignments, ensuring mission commands are authentic and untamperable. ML-KEM secures task payloads, protecting the confidentiality of mission-specific data during transmission.	M18
PQ-TLS	The crypto primitives provided as part of the PiQASO Software Library are used as part of higher-order cryptographic protocols. In the context of the Automation use case, PQ-TLS is used in order to establish a secure communication channel between the FMA and the Autonomous Mobile Robots (AMRs) using quantum-resistant handshakes. In addition, in the context of the Automation use case, Authenticated Key Exchange (AKE) is used for establishing ephemeral sessions for secure robot-to-robot interactions.	M18
PQC-as-a-Service (PQCaaS)	The PiQASO PQCaaS component is able to provide post-quantum crypto functionalities to devices which may not have the computational capabilities to do so on their own. As described in detail in D3.1, the PQCaaS component offers the capability for data encryption at the side of the PQ Data Encryption Server, so that such devices do not need to perform the encryption themselves.	M36
Updatable Public Key Encryption (UPKE)	The UPKE capability of PiQASO, offered by the PQCaaS modality, is a notion of asymmetric encryption that has been demonstrated to be an important tool for key rotation in outsourced storage, and can also be constructed from quantum-safe assumptions. In the context of the Automation use case, this is leveraged for performing direct encryption of operational data by the AMRs, allowing robots to project telemetry before transmission or storage.	M36
Attribute-Based Encryption (ABE)	In order to provide granular attribute-based access, the ABE modality of the PQCaaS component offers the capability for lattice-based encryption leveraging PQ crypto primitives, so that the encryption server can encrypt a set of data under a specific set of attributes. In the Automation use case, this is applied to operational logging to ensure that rosbags and maps are accessible only to users with specific roles (e.g., Administrators).	M36
Hardware Acceleration	In the context of the Automation use case, the FPGA-based hardware acceleration capabilities of PiQASO are leveraged in order to offload PQC primitives to modules with the capability to perform these operations, ensuring real-time navigation performance.	M36

Table 16.3: Functionalities for the Finance use case

Functionality	Description	Status
QR-TCB	The Quantum-Resistant Trusted Computing Base (QR-TCB) of PiQASO is designed as the minimal trusted domain responsible for protecting long-term cryptographic assets and enforcing secure execution of Post-Quantum Cryptography (PQC). It is required in order to support the crypto capabilities detailed throughout this Table.	M18
PiQASO Software Library	The PiQASO Software Library may enable the use of PQC primitives, such as ML-KEM and ML-DSA, in devices that may support such operations. However, even if internal systems adopt PQC, the final leg of communication with external clients remains vulnerable if the edge hardware cannot process PQC. This security gap can also be addressed through the PQC offerings of the PiQASO Software Library.	M18

PQ-TLS	PiQASO enables secure online banking operations by integrating Post-Quantum Cryptography (PQC) into the external communication flow, and specifically PQ-TLS to govern interactions between customers and the bank backend infrastructure, while maintaining compatibility with existing RSA-based internal systems. This hybrid approach allows CXB to achieve quantum-safe communications without requiring immediate hardware replacement or full infrastructure overhaul.	M18
PQC-as-a-Service (PQCaaS)	PQC algorithms are computationally intensive and may introduce latency in real-time payment flows. However, several critical components in the current CXB infrastructure—such as the SSL Load Balancer (Thunder 5840S-11), external firewalls (SRX-5400), and routers/switches—do not natively support PQC algorithms. Replacing these devices is not feasible in the short term due to cost, operational risk, and vendor dependencies. To this end, the PiQASO PQCaaS component is able to provide post-quantum crypto functionalities to devices which may not have the computational capabilities to do so on their own. This approach offloads cryptographic operations to a secure backend, enabling quantum-safe communications without requiring immediate hardware upgrades.	M36
Updatable Public Key Encryption (UPKE)	The introduction of new key types and exchange mechanisms requires a re-thinking of key lifecycle management, including generation, distribution, rotation, and revocation. Existing systems are not designed to handle these complexities. The PiQASO UPKE scheme has been demonstrated to be an important tool for key rotation in outsourced storage, and can also be constructed from quantum-safe assumptions. It offers forward security through regular updates of public/private key pairs such that potential compromise of future private keys does not break confidentiality of ciphertexts encrypted under earlier public keys.	M36
Hardware Acceleration	PiQASO must ensure that the integration of PQC does not degrade user experience or violate service-level agreements. To this end, aside from the PQCaaS approach outline above, the Finance use case can leverage the Hardware Acceleration capabilities in order to execute PQC operations while also fulfilling the performance and latency requirements of the system.	M36

Table 16.4: Functionalities for the Energy use case

Functionality	Description and Purpose in Energy Use Case	Status
QR-TCB	Protects long-term PQC cryptographic assets and enforces secure PQC execution on MOH substation gateways and MORE edge gateways. Foundational enabler for all PQC operations in UC4.	M18
PiQASO Software Library (ML-KEM / ML-DSA)	Provides optimised implementations of ML-KEM and ML-DSA for use on gateway nodes at MOH (substation gateways) and MORE (edge gateways). Enables local PQC signing and key encapsulation for PQ-TLS sessions and telemetry integrity.	M18
PQ-TLS	Quantum-resistant TLS sessions using ML-KEM for key exchange and ML-DSA for mutual authentication. Used for SCADA-to-gateway communications at MOH and for gateway-to-MOH channels at MORE.	M18
PQCaaS Backend	Cloud-based PQC-as-a-Service for offloading cryptographic operations from constrained devices (MORE field sensors, legacy SCADA components). Also provides certificate management and algorithm transition support.	M36
Attribute-Based Encryption (ABE)	Policy-driven encryption of energy data records, ensuring that only authorised roles (e.g., MOH analysts, CERTH monitoring) can access specific data categories. Deployed at MORE gateways and MOH data ingestion layer.	M36
Functional Encryption / Updatable Public Key Encryption (UPKE)	Enables homomorphic and function-specific computations on encrypted energy telemetry. Applied in Scenario #2 (MORE) for privacy-preserving load forecasting and anomaly detection without exposing raw production data.	M36

Crypto Assessment Toolkit / Trust Assessment Framework (TAF)	Monitors the cryptographic posture of the UC4 deployment, triggers algorithm transitions (crypto-agility), and provides tamper-evident audit logs. Integrated with the CERTH monitoring infrastructure.	M18
Crypto-Agility Framework	Runtime mechanism enabling dynamic algorithm suite transitions on gateway nodes without service interruption. Required for Scenario #2 (MOH) and ensures long-term quantum readiness across both MOH and MORE deployments.	M36
Secure Bootstrapping / Device Onboarding	PiQASO lightweight bootstrapping protocol for onboarding constrained IoT/OT devices at MORE production sites. Coordinates between the edge gateway, QR-TCB, and PQCaaS Backend to provision PQC credentials.	M18

Table 16.5: Functionalities for the Healthcare use case

Functionality	Description	Status
QR-TCB	The Quantum-Resistant Trusted Computing Base (QR-TCB) of PiQASO is designed as the minimal trusted domain responsible for protecting long-term cryptographic assets and enforcing secure execution of Post-Quantum Cryptography (PQC). It is required in order to support the crypto capabilities detailed throughout this Table.	M18
PiQASO Software Library	In the Healthcare use case, the secure transmission of data is considered as a core necessity. This applies to both modalities of this use case: (i) In the <i>Remote Patient Monitoring Intelligent System for Supporting Independent and Safe Living</i> case, data from medical devices deployed at the patient's home is transmitted to the hospital or healthcare organization backend in order to provide remote patient care, and (ii) in the <i>Smart Ambulance</i> case, data is transmitted from the ambulance module to the hospital backend in order to facilitate healthcare provision during transit. Thus, it is evident that the establishment of secure and authenticated communication channels is required. To this end, the PiQASO Software Library provides the ML-KEM and ML-DSA crypto enablers, highly optimized in order to operate on various types of embedded devices used in the context of this use case.	M18
PQC-as-a-Service (PQCaaS)	As previously detailed, the PQCaaS component of PiQASO is intended to provide PQ crypto capabilities to devices that may not have the computational capabilities to perform such operations themselves, such as the low-end medical devices encountered in this use case, in an as-a-service manner. In this case, Functional Encryption operates as a modality of the PQCaaS component of PiQASO.	M36
Functional Encryption	As detailed in D3.1, the Functional Encryption capability of PiQASO enables carrying out computations on encrypted data, revealing only the result of these computations and avoid leakage of any sensitive information. In the context of the use case, the common denominator is the need to share data in the backend service of the healthcare delivery organization (e.g., hospital), in order to enable monitoring the treatment of the patient. For instance, this may apply to ECG data from monitoring the heart rate of a patient. In this case, Functional Encryption can be used in order to determine whether any recorded values match or exceed a threshold value, in order to determine whether an alarm should be raised signifying the need for further medical attention.	M36

Table 16.6: Functionalities for the Aerospace use case

Functionality	Description	Status
---------------	-------------	--------

QR-TCB	The Quantum-Resistant Trusted Computing Base (QR-TCB) of PiQASO is designed as the minimal trusted domain responsible for protecting long-term cryptographic assets and enforcing secure execution of Post-Quantum Cryptography (PQC). It is required in order to support the crypto capabilities detailed throughout this Table. We also leverage FPGA-based modules with the QR-TCB as transparent network proxies in order to offload heavy PQC operations, ensuring cryptographic processing does not interfere with the SUT's simulated flight dynamics or trigger the need for expensive hardware recertification.	test
PiQASO Software Library	The PiQASO Software Library offers highly optimized implementations of the standardized ML-KEM and ML-DSA NIST-standardized schemes, which are used in order to establish secure and authenticated communication channels. In the context of the Aerospace use case, the integration of post-quantum functionalities into MQTT, DDS, and OPCUA exchanges bridges the gap between rigorous cryptographic security and sub-100ms timing constraints. This includes both the aforementioned PQC schemes whose optimization aids the fulfilment of these timing constraints, as well as their application into higher-end protocols (e.g., PQ-TLS), for the establishment of secure and authenticated communication channels.	M18
PQC-as-a-Service (PQCaaS)	The PiQASO PQCaaS component is able to provide post-quantum crypto functionalities to devices which may not have the computational capabilities to do so on their own. In the context of the Aerospace use case, which is subject to strict latency constraints of slat flap control systems with control response times bounded strictly under 100ms, the PQCaaS component can facilitate the fast execution of PQC so they do not need to be performed on components with low computational capabilities.	M36
Attribute-Based Encryption (ABE), Updatable Public Key Encryption (UPKE)	An important requirement of the Aerospace use case is Intellectual Property (IP) protection, as aerospace control algorithms represent decades of costly research and development and need to be protected from unauthorized personnel. Unencrypted telemetry, test parameters, and system logs transmitted across the testing network are also highly susceptible to espionage, exposing core competitive advantages to nation-state actors or industry competitors. To this end, the UPKE scheme can provide advanced encryption capabilities for stored data-at rest, while the ABE scheme can provide identity management capabilities for SUTs and virtual simulators across geographically separated testing nodes.	M36
Hardware Acceleration	As aforementioned, the Aerospace use case is subject to strict timing and latency constraints. The hardware acceleration capabilities of PiQASO are able to assist in the fulfilment of these constraints, by offloading heavy PQC operations to FPGA-based transparent network proxies, which are then able to efficiently perform the underlying mathematical computations.	M36

Table 16.7: Functionalities for the Online Media use case

Functionality	Description	Status
QR-TCB	Preparing DPG Media's editorial infrastructure to handle the evolving cybersecurity threats posed by quantum computers, thus securing long-term operational continuity. The Quantum-Resistant Trusted Computing Base (QR-TCB) of PiQASO works towards this goal, as it is responsible for protecting long-term cryptographic assets and enforcing secure execution of Post-Quantum Cryptography (PQC). It is required in order to support the crypto capabilities detailed throughout this Table.	M18

PiQASO Software Library	A core requirement of the Online Media use case entails securely enabling journalists to authenticate and upload articles to DPG Media's main platform in a quantum-resistant manner, ensuring the integrity, confidentiality, and availability of sensitive content throughout the editorial process. Specifically, as part of this process, a journalist logs in to the CMS using predefined credentials, a process that should be protected by quantum-resistant mechanisms e.g. lattice-based KEMs. This is offered by the PiQASO Software Library, which contains highly optimized implementations of the standardized ML-KEM and ML-DSA NIST-standardized schemes, which are used in order to establish secure and authenticated communication channels.	M18
PQ-TLS	The PQC schemes of the PiQASO Software Library are also used in the context of higher order protocols, such as PQ-TLS. In the context of the Online Media use case, data integrity and confidentiality have to be maintained throughout transmission. Thus, as part of the secure article upload process, articles will be uploaded to the CMS over a PQ-TLS connection, using hybrid encryption (e.g., AES combined with ML-KEM-based key exchange). As part of the Editorial Review and Publication, process, editors securely access the CMS for reviewing and approving articles. Once finalized, the article will be published on DPG Media's public-facing portals via secure a secure communication (PQ-TLS).	M18
PQC-as-a-Service (PQCaaS)	The PiQASO PQCaaS component is able to provide post-quantum crypto functionalities to devices which may not have the computational capabilities to do so on their own. It provides the ABE and UPKE functionalities, which will be analysed in the following.	M36
Updatable Public Key Encryption (UPKE), Attribute-Based Encryption (ABE)	In the context of the Online Media use case, Uploaded articles must be encrypted at rest employing PQC primitives, and role-based access control should ensure that only authorized editors or team members can retrieve or modify the content. This is achieved through the Forward Encryption capabilities offered by the UPKE scheme of PiQASO, which has been demonstrated to be an important tool for key rotation in outsourced storage, and can also be constructed from quantum-safe assumptions. Additionally, the ABE scheme is able to offer the required highly granular access control, so that only entities with the required attributes are able to access sets of content.	M36

Table 16.8: Functionalities for the UAV use case

Functionality	Description	Status
QR-TCB	The Quantum-Resistant Trusted Computing Base (QR-TCB) of PiQASO is designed as the minimal trusted domain responsible for protecting long-term cryptographic assets and enforcing secure execution of Post-Quantum Cryptography (PQC). It is required in order to support the crypto capabilities detailed throughout this Table. By offloading complex PQC computations to dedicated hardware, PiQASO maintains low latency and high throughput on the secure link. This means the encryption/authentication overhead does not bog down the UAV's control responsiveness – an important factor in counter-jamming or DoS.	M18
PiQASO Software Library	The PiQASO Software Library secures the UAV-ground station link by providing advanced PQC for encryption and key exchange (ML-KEM, ML-DSA implementations). By establishing a quantum secure communication channel (e.g., through a PQ-TLS handshake), PiQASO ensures that all command-and-control and telemetry data is strongly encrypted and authenticated. The integrity of the channel is guaranteed through PQ digital signatures (ML-DSA) allowing the UAV and ground station to perform mutual verification, and verification of exchanged messages.	M18

PQC-as-a-Service (PQCaaS)	PiQASO's supports scenarios with multiple ground stations or UAV swarms, which are often more complex security-wise, through the PQC-as-a-Service model, through which the framework can act as a central trust anchor for many nodes. For example, if multiple ground control stations need to communicate with one drone (or multiple drones), PiQASO's cloud-based key management service (PQ-DS) can distribute quantum-safe keys and certificates to all parties. In addition, such a unified PQC service allows consistent security updates and posture across all units, for instance by rolling out a new QR algorithm through the service to all drones and stations simultaneously.	M36
Updatable Public Key Encryption (UPKE)	The UAV use case entails the need for transmitting and securely storing sensitive mission-critical data, such as flight commands, telemetry reports, or images. As aforementioned, the PQCaaS modality of PiQASO provides the UPKE scheme, which is an important tool for key rotation in outsourced storage, constructed from quantum-safe assumptions. It offers forward security through regular updates of public/private key pairs such that potential compromise of future private keys does not break confidentiality of ciphertexts encrypted under earlier public keys. Also, for the transmission of such data, it enables the derivation of multiple unlinkable private keys from a single public key, thus ensuring that if a specific UAV is compromised or tampered with, this will not affect the data transmitted or received by other UAV modules.	M36
Attribute-Based Encryption (ABE)	In the context of the UAV use case, it is imperative to ensure that only trusted entities can interact with the UAV. To this end, PiQASO employs fine-grained access control policies – for example, using ABE and tokens so that even within a mission, each data payload or command can require specific authorization. This directly counters weak authentication vulnerabilities and spoofing threats, so that an attacker cannot impersonate the ground station because they lack the PQ-certified keys and tokens, and any unauthorized command will fail verification.	M36
Hardware Acceleration	As aforementioned, in order to maintain low latency and high throughput on the secure link, complex PQC computations can be offloaded to dedicated hardware. As the hardware optimizations of PiQASO offer the capability of executing PQC in a timely and efficient manner, the channel can use spread-spectrum or rapid re-keying without undue delay, helping to mitigate jamming by quickly switching frequencies or sessions if an attack is detected. Furthermore, PiQASO's comprehensive testing in resource-constrained environments ensures reliability when the drone is under stress.	M36

Table 16.9: Functionalities for the Transportation use case

Functionality	Description	Status
QR-TCB	The Quantum-Resistant Trusted Computing Base (QR-TCB) of PiQASO is designed as the minimal trusted domain responsible for protecting long-term cryptographic assets and enforcing secure execution of Post-Quantum Cryptography (PQC). It is required in order to support the crypto capabilities detailed throughout this Table. In the Transportation use case, it is used for storing keys in drones, as drones may be physically captured or inspected by adversaries. Thus, the QR-TCB offers the capability for hardware-backed keys (secure storage and, where possible, hardware-assisted cryptographic operations) to reduce the risk of secret-key extraction and identity cloning.	M18
PiQASO Software Library	Secure mission configuration is a critical requirement of the Transportation use case, as mission packages contain sensitive operational details (e.g., routes, inspection targets, and constraints) and directly influence safety and compliance. The pipeline therefore needs strong authenticity, integrity, and confidentiality so that only authorized operators can deploy missions, drones can verify mission provenance, and attackers cannot tamper with or replay mission instructions. The PiQASO Software Library provides highly optimized PQC implementations, such as ML-KEM for establishing secure and authenticated channels, and ML-DSA which can be used by the operator to sign mission plans.	M18

PQ-TLS	The PQ-TLS protocol can be used in order to establish a secure communication between various endpoints. Specifically, as part of the operation of the Transportation use case, when creating a mission package, the drone operator defines a new inspection mission (e.g., route, altitude profile, and expected inspection segments) and uploads it to the GCS. Then, the GCS invokes the PiQASO SDK in order to sign the mission package using a PQ signing scheme, and establish a PQ-TLS session with a drone. For secure mission deployment, the signed mission package is transmitted to the on-board unit (OBU) via the GCS over the established PQ-TLS channel, and the OBU uses its QR-TCB capability to verify the PQ signature and confirm that the package originates from an authorized mission planner. Only if all verifications succeed does the OBU accept and activate the mission.	M18
PQC-as-a-Service (PQCaaS)	For fulfilling the performance and latency requirements of the Transportation use case, cryptographic operations must be implemented efficiently to avoid excessive CPU load, memory footprint, latency, and energy consumption while still providing strong security guarantees. This is particularly pertinent in cases where PQ algorithms require large key sizes and induce computational overhead, and when secure channels and signing must be used routinely during mission deployment and anomaly reporting. To this end, the PQCaaS offers PQ capabilities in an as-a-service manner, in order to enable offloading such complex operations to the backend and fulfilling the aforementioned requirements.	M36
Attribute-Based Encryption (ABE)	A core requirement of the Transportation case entails strengthening the authenticity and integrity of anomaly reporting. In this regard, the backend needs high assurance that an anomaly report originates from a legitimate drone executing an approved mission and that the reported evidence has not been altered in transit. This requires robust signing and verification workflows and careful identity management, including credential provisioning and lifecycle handling for operators, drones, and services. To this end, the ABE functionality is offered through the PQCaaS component of PiQASO, in order to enable the encryption of anomaly reports in a manner that ensures that only entities with the appropriate credentials can access them.	M36
Hardware Acceleration	As aforementioned, the resource-constrained nature of UAV platforms and on-board units necessitates the efficient implementation of PQC operations to avoid excessive CPU load, memory footprint, latency, and energy consumption. To this end, the hardware acceleration capabilities of PiQASO lead to the faster computation of the underlying mathematical operations, thus leading to the more efficient execution of PQC operations.	M36

Chapter 17

Summary and Conclusions

As previously detailed, one core target of PiQASO is to *provide the tools required for the transition of such SGCs to the post-quantum era, through the provision of the required tools and security solutions for ensuring that all entities remain at a trustworthy level throughout their operational lifecycle*. One important challenge in this regard is that existing service graph chains (SGCs) may consist of heterogeneous devices with varying computational capabilities, including legacy devices with limited ability to support the advanced mathematical capabilities required for executing complex post-quantum cryptographic operations. To this end, D2.1 provided a comprehensive overview of the PiQASO architecture and its application to the post-quantum transition of a broad and heterogeneous set of service graph chains. Building upon the technical foundations and PQC migration rationale established in D3.1, the deliverable has defined a structured architecture that positions and integrates PiQASO components across diverse target infrastructures, and has detailed the post-quantum transition pathways for nine application domains: Automotive, Automation, Finance, Energy, Healthcare, Aerospace, Online Media, UAVs, and Transportation.

A central outcome of this deliverable is the systematic requirements elicitation performed for each use case, following the top-down, defence-driven methodology presented in Chapter 3. This approach has ensured that the identified requirements are grounded in operational reality and reflect the specific constraints and threat models of each domain, ranging from the strict sub-100ms latency constraints of vehicular safety and aerospace control systems, to the resource-limited nature of embedded medical devices and UAV platforms, to the long-term data confidentiality requirements of financial and energy infrastructures. For each domain, a set of Reference Scenarios and User Stories has been defined, providing a concrete and structured basis for the experimentation activities that will be conducted through the PiQASO testbed in subsequent project phases.

The analysis of the PiQASO PQC Ensemble functionalities required across all use cases, presented in Chapter 16, reveals a consistent set of core capabilities that underpin the post-quantum transition across domains. The **Quantum-Resistant Trusted Computing Base (QR-TCB)** emerges as a universal foundational requirement, serving as the minimal trusted domain for protecting long-term cryptographic assets and enforcing the secure execution of PQC across all eleven (11) use cases, representing nine (9) industrial domains. The **PiQASO SDK**, providing highly optimized implementations of the NIST-standardized ML-KEM and ML-DSA schemes, is equally pervasive, enabling the establishment of secure and authenticated communication channels across heterogeneous endpoints, from vehicle on-board units and autonomous mobile robots to medical devices and drone platforms. The **PQ-TLS** protocol, leveraging these primitives, is identified as the primary mechanism for securing data-in-transit across the Automation, Finance, Aerospace, Online Media, and Transportation use cases.

Beyond these foundational capabilities, the deliverable highlights the central role of the **PQC-as-a-Service**

(PQCaaS) component in enabling the post-quantum transition of constrained and legacy devices that cannot perform PQC operations natively. This modality is identified as a requirement across all eleven use cases, and serves as the delivery mechanism for advanced capabilities including **Updatable Public Key Encryption (UPKE)**, which provides forward secrecy through quantum-safe key rotation for outsourced storage environments, and **Attribute-Based Encryption (ABE)**, which enables fine-grained, policy-driven access control over sensitive data across the Automation, Energy, Aerospace, Online Media, UAV, and Transportation domains. The **Functional Encryption** capability, which presents particular applicability in the Healthcare use case, further extends the PiQASO offering by enabling privacy-preserving computations over encrypted data, a particularly pertinent requirement in domains where sensitive telemetry or patient data must be processed without exposing personally identifiable patient information. Finally, **Hardware Acceleration** through FPGA-based modules is identified as a critical enabler for meeting the stringent latency and throughput requirements of multiple use cases, ensuring that the computational overhead of PQC algorithms does not compromise operational performance.

Overall, the work presented in D2.1 establishes a well-structured and operationally grounded foundation for the subsequent phases of the PiQASO project. The defined architecture, requirements, Reference Scenarios, User Stories, and functionality mappings provide the necessary inputs for the design and implementation of the PiQASO testbed, and will directly govern the experimentation and evaluation activities planned across all demonstrators. In doing so, D2.1 charts a coherent and systematic path from the architectural vision of PiQASO towards its practical validation in the context of real-world, mission-critical service graph chains.

Bibliography

- [1] Maldrone: Malware which hijacks your personal drone. ZDNET.
- [2] OWASP top 10 drone security risks. OWASP Foundation.
- [3] S. M. Albrektsen, T. H. Bryne, and T. A. Johansen. Robust and secure UAV navigation using GNSS, phased-array radio system and inertial sensor fusion. In 2018 IEEE Conference on Control Technology and Applications (CCTA), pages 1338–1345, Copenhagen, Denmark, 2018.
- [4] O. M. Alhawi, M. A. Mustafa, and L. C. Cordiro. Finding security vulnerabilities in unmanned aerial vehicles using software verification. In 2019 International Workshop on Secure Internet of Things (SIOT), pages 1–9, Luxembourg, Luxembourg, 2019.
- [5] Alias Robotics. Cybersecurity in the ROS 2 and the data distribution service (DDS): Vulnerabilities and mitigations. <https://news.aliasrobotics.com/cybersecurity-in-the-ros-2-communication-middleware-targeting-the-top-6-dds-implementations/>, 2022. Accessed: 2026-07-14.
- [6] T. Alladi, Naren, G. Bansal, V. Chamola, and M. Guizani. SecAuthUAV: A novel authentication scheme for UAV-ground station and UAV-UAV communication. IEEE Transactions on Vehicular Technology, 69(12):15068–15077, December 2020.
- [7] Azza Allouch, Omar Cheikhrouhou, Anis Koubaa, Mohamed Khargui, and Tarek Abbes. MAVSec: Securing the MAVLink protocol for Ardupilot/PX4 unmanned aerial systems. In 2019 International Wireless Communications and Mobile Computing Conference (IWCMC), pages 621–628, 2019.
- [8] Wiam Alluhybi and Omar Alhazmi. Towards a threat model for unmanned aerial vehicles. 2021.
- [9] Amazon. Amazon deploys over 1 million robots and launches new AI foundation model. <https://www.aboutamazon.com/news/operations/amazon-million-robots-ai-foundation-model>, 2025. Accessed: 2026-07-14.
- [10] Arthur Amorim, Max Taylor, Trevor Kann, Gary Leavens, William Harrison, and Lance Joneckis. UAV resilience against stealthy attacks. In 2025 International Conference on Unmanned Aircraft Systems (ICUAS), pages 994–1001, 2025.
- [11] Martin Andreoni, Michael Baddeley, Willian T. Lunardi, Anshul Pandey, and Jean-Pierre Giacalone. Towards secure wireless mesh networks for UAV swarm connectivity: Current threats, research, and opportunities, 2021.
- [12] I. Astaburuaga, A. Lombardi, B. La Torre, C. Hughes, and S. Sengupta. Vulnerability analysis of AR.Drone 2.0, an embedded linux system. In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pages 0666–0672, Las Vegas, NV, USA, 2019.

- [13] Elaine Barker, Tim Polk, and Murugiah Souppaya. Considerations for Achieving Cryptographic Agility. <https://csrc.nist.gov/pubs/cswp/39/considerations-for-achieving-cryptographic-agility/final>, 2025. NIST Cybersecurity White Paper 39. Accessed: 2026-05-02.
- [14] J. Blanco-Romero, F. Almenares Mendoza, C. Garcia Rubio, C. Campo, and D. Diaz Sanchez. On the practical feasibility of harvest-now, decrypt-later attacks, 2026.
- [15] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on post-quantum cryptography. Technical Report NISTIR 8105, National Institute of Standards and Technology (NIST), 2016.
- [16] G. Choudhary, V. Sharma, I. You, K. Yim, I.-R. Chen, and J.-H. Cho. Intrusion detection systems for networked unmanned aerial vehicles: A survey. In 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), pages 560–565, Limassol, Cyprus, 2018.
- [17] G. Choudhary, V. Sharma, I. You, K. Yim, I.-R. Chen, and J.-H. Cho. Intrusion detection systems for networked unmanned aerial vehicles: A survey. In 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), pages 560–565, Limassol, Cyprus, 2018.
- [18] Mike Cohn. User stories applied: For agile software development. Addison-Wesley Professional, 2004.
- [19] Cybersecurity and Infrastructure Security Agency. Software Bill of Materials (SBOM). <https://www.cisa.gov/sbom>, 2025. Accessed: 2026-05-02.
- [20] Cybersecurity and Infrastructure Security Agency (CISA). ICS advisory (ICSA-18-191-01): Universal robots robot controllers. <https://www.cisa.gov/news-events/ics-advisories/icsa-18-191-01>, 2018. Accessed: 2026-07-14.
- [21] Drew Davidson, Hao Wu, Robert Jellinek, Thomas Ristenpart, and Vikas Singh. Controlling UAVs with sensor input spoofing attacks. In Proceedings of the 10th USENIX Conference on Offensive Technologies, WOOT’16, pages 221–231, USA, 2016. USENIX Association.
- [22] Wenyu Dong, Guangquan Xu, Shicheng Feng, Hongpeng Bai, Michael Sheng, and Xi Zheng. Escorting the confidentiality and integrity of uavs: What exactly can trusted execution environment offer? ACM Comput. Surv., 58(3), September 2025.
- [23] Wenyu Dong, Guangquan Xu, Shicheng Feng, Hongpeng Bai, Quan Sheng, and Xi Zheng. Escorting the confidentiality and integrity of UAVs: What exactly can trusted execution environment offer? ACM Computing Surveys, 58, 2025.
- [24] S. Falas, C. Konstantinou, and M. K. Michael. Hardware-enabled secure firmware updates in embedded systems. In C. Metzler, P. E. Gaillardon, G. De Micheli, C. Silva-Cardenas, and R. Reis, editors, VLSI-SoC: New Technology Enabler, volume 586 of IFIP Advances in Information and Communication Technology. Springer, Cham, 2020.
- [25] Tobias Frauenschläger and Jürgen Mottok. Problems and New Approaches for Crypto-Agility in Operational Technology. In 12th European Congress Embedded Real Time Systems - ERTS 2024, Toulouse, France, June 2024.
- [26] Andrii Grekhov, Volodymyr Kharchenko, and Vasyl Kondratiuk. Deep Q-network based resilient drone communication: Neutralizing first-order markov jammers, 2026.

- [27] Hassan Jalil Hadi, Yue Cao, Khaleeq Un Nisa, Abdul Majid Jamil, and Qiang Ni. A comprehensive survey on security, privacy issues and emerging defence technologies for UAVs. Journal of Network and Computer Applications, 213:103607, 2023.
- [28] R. H. Jacobsen and A. Marandi. Security threats analysis of the unmanned aerial vehicle system. In MILCOM 2021 – 2021 IEEE Military Communications Conference (MILCOM), pages 316–322, San Diego, CA, USA, 2021.
- [29] Jinseob Jeong, Dongkwan Kim, JoonHa Jang, Juhwan Noh, Changhun Song, and Yongdae Kim. Un-rocking drones: Foundations of acoustic injection attacks and recovery thereof. In Network and Distributed System Security Symposium (NDSS), 2023.
- [30] S. Z. Khan, M. Mohsin, and W. Iqbal. On GPS spoofing of aerial platforms: A review of threats, challenges, methodologies, and future research directions. PeerJ Computer Science, 7:e507, May 2021.
- [31] Samantha Klier and Harald Baier. Beware of the rabbit hole—a digital forensic case study of DIY drones. In Secure IT Systems: 29th Nordic Conference, NordSec 2024, Karlstad, Sweden, November 6–7, 2024, Proceedings, pages 325–344. Springer Verlag, Berlin, Heidelberg, 2025.
- [32] Yasaswini Konapalli, Lotfi Othmane, Cihan Tunc, Feras Benchellal, and Likhita Mudagere. Reverse engineering and control-aware security analysis of the ArduPilot UAV framework, 2025.
- [33] Naveen Kumar and Ankit Chaudhary. Litewtaka: Authenticating uav-gcs and uav–uav communication using secure and lightweight mechanism based on puf. Journal of Network and Computer Applications, 245:104372, 2026.
- [34] E. Kurtpinar. Privacy’s sky-high battle: The use of unmanned aircraft systems for law enforcement in the european union. Journal of Intelligent Robotic Systems, 110, 2024.
- [35] Y.-M. Kwon, J. Yu, B.-M. Cho, Y. Eun, and K.-J. Park. Empirical analysis of MAVLink protocol vulnerability for attacking unmanned aerial vehicles. IEEE Access, 6:43203–43212, 2018.
- [36] Young-Min Kwon, Jaemin Yu, Byeong-Moon Cho, Yongsoon Eun, and Kyung-Joon Park. Empirical analysis of MAVLink protocol vulnerability for unmanned aerial vehicles. IEEE Access, pages 1–1, 2018.
- [37] Dasom Lee, David J. Hess, and Michiel A. Heldeweg. Safety and privacy regulations for unmanned aerial vehicles: A multiple comparative analysis. Technology in Society, 71:102079, 2022.
- [38] T. Liu, G. S. Ramachandran, and R. Jurdak. Post-quantum cryptography for internet of things: A survey on performance and optimization, 2024.
- [39] Jesus Lopez, Mohammad Saidur Rahman, et al. Evaluating post-quantum cryptographic algorithms on resource-constrained devices. <https://arxiv.org/abs/2507.08312>, 2025.
- [40] Simegneu Meheretu, Ethiopia Nigussie, and Gebeyehu Gebremeskel. A systematic literature review on spoofing and jamming approaches in unmanned aerial vehicles navigation. Journal of Aerospace Technology and Management, 17, 2025.
- [41] Yassine Mekdad, Ahmet Aris, Leonardo Babun, Abdeslam El Fergougui, Mauro Conti, Riccardo Lazzeretti, and A. Selcuk Uluagac. A survey on security and privacy issues of uavs. Computer Networks, 224:109626, 2023.

- [42] Yassine Mekdad, Ahmet Aris, Leonardo Babun, Abdeslam El Fergougui, Mauro Conti, Riccardo Lazzeretti, and A. Selcuk Uluagac. A survey on security and privacy issues of UAVs. Computer Networks, 224:109626, 2023.
- [43] Yassine Mekdad, Ahmet Aris, Leonardo Babun, Abdeslam El Fergougui, Mauro Conti, Riccardo Lazzeretti, and A. Selcuk Uluagac. A survey on security and privacy issues of UAVs. Computer Networks, 224:109626, 2023.
- [44] Mindray. BeneVision N1 monitor/module: Detailed specifications. <https://www.mindray.com/na/products/patient-monitoring/continuous-patient-monitoring/high-acuity/benevision-n1/>, 2024. Accessed: 2026-07-14.
- [45] M. Mosca. Cybersecurity in an era with quantum computers: Will we be ready? IEEE Security Privacy, 16(5):38–41, 2018. September/October.
- [46] Michele Mosca. Cybersecurity in an era with quantum computers: Will we be ready? IEEE Security & Privacy, 16(5):38–41, 2018.
- [47] B. Nassi, R. Ben-Netanel, A. Shamir, and Y. Elovici. Drones’ cryptanalysis-smashing cryptography with a flicker. In 2019 IEEE Symposium on Security and Privacy (SP), pages 1397–1414. IEEE Computer Society, 2019.
- [48] National Cybersecurity Center of Excellence (NCCoE). Migration to Post-Quantum Cryptography. <https://www.nccoe.nist.gov/applied-cryptography/migration-to-pqc>, 2024. Accessed: 2026-05-02.
- [49] National Institute of Standards and Technology. Nist announces first four quantum-resistant cryptographic algorithms. Technical report, NIST, 2022. Accessed: 2025-07-07.
- [50] National Institute of Standards and Technology (NIST). Post-quantum cryptography standardization. <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization>, 2024. Accessed: 2026-07-14.
- [51] National Institute of Standards and Technology (NIST). Transition to post-quantum cryptography standards (NIST IR 8547). Technical report, National Institute of Standards and Technology, 2024. Quantum computing threat timeline and cryptographic algorithm deprecation schedule.
- [52] Office of Management and Budget. Memorandum M-23-02: Migrating to Post-Quantum Cryptography. <https://www.whitehouse.gov/wp-content/uploads/2022/11/M-23-02-M-Memo-on-Migrating-to-Post-Quantum-Cryptography.pdf>, November 2022. Accessed: 2026-05-02.
- [53] Open Source Robotics Foundation / ROS 2 Design. ROS 2 DDS-security integration. https://design.ros2.org/articles/ros2_dds_security.html, 2024. Accessed: 2026-07-14.
- [54] OWASP CycloneDX. Cryptography Bill of Materials (CBOM). <https://cyclonedx.org/capabilities/cbom/>, 2024. Accessed: 2026-05-02.
- [55] PiQASO. D3.1 PiQASO QR Crypto Primitives Design, Optimization and Acceleration. Deliverable D3.1, The PiQASO Consortium, 15 2026.
- [56] PiQASO. D3.2 PiQASO SDK and PQC as-a-service – Initial Version. Deliverable D3.2, The PiQASO Consortium, 18 2026.

- [57] PIQASO Consortium. Crypto Asset Inventory Repository. https://gitlab.com/piqaso/crypto_asset_inventory, 2026. Project software artefact. Access may be subject to project-level repository permissions. Accessed: 2026-05-02.
- [58] H. Pirayesh and H. Zeng. Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 24(2):767–809, 2022. Second quarter.
- [59] C. Profentzas, M. Gunes, Y. Nikolakopoulos, O. Landsiedel, and M. Almgren. Performance of secure boot in embedded systems. In *2019 15th International Conference on Distributed Computing in Sensor Systems, DCOSS*, pages 198–204, Santorini, Greece, 2019.
- [60] Joshua Renckens, Peter B. Rønne, Johann Großschädl, and Peter Y. A. Ryan. An evaluation of post-quantum and hybrid noise protocol variants on mobile devices. In *Innovative Security Solutions for Information Technology and Communications: 17th International Conference, SecITC 2024, Bucharest, Romania, November 21–22, 2024, Revised Selected Papers*, page 149–169, Berlin, Heidelberg, 2024. Springer-Verlag.
- [61] Rodrigo Roman, Javier Lopez, and Masahiro Mambo. Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Computer Systems*, 78:680–698, 2018.
- [62] Alessio Rugo, Claudio Ardagna, and Nabil El Ioini. A security review in the UAVNet era: Threats, countermeasures, and gap analysis. *ACM Computing Surveys*, 55:1–35, 2022.
- [63] Harshad Sathaye, Martin Strohmeier, Vincent Lenders, and Aanjhan Ranganathan. An experimental study of GPS spoofing and takeover attacks on UAVs. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3503–3520, 2022.
- [64] Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum tls without handshake signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, CCS '20*, pages 1461–1480, New York, NY, USA, 2020. Association for Computing Machinery.
- [65] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 124–134. IEEE, 1994.
- [66] Alexandr Silonosov and Lawrence Henesey. Telemetry data sharing based on attribute-based encryption schemes for cloud-based drone management system. pages 1–8, 2024.
- [67] Simone Soderi, Daniele Masti, and Yuriy Zacchia Lun. Railway cyber-security in the era of interconnected systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 24(7):6764–6779, 2023.
- [68] Supply Chain Today. Autonomous robots revolutionizing supply chain. <https://www.supplychaintoday.com/autonomous-robots-revolutionizing-supply-chain/>, 2024. Accessed: 2026-07-14.
- [69] Greg Thornton and Pooneh Bagheri Zadeh. An investigation into unmanned aerial system (UAS) forensics: Data extraction analysis. *Forensic Science International: Digital Investigation*, 41:301379, 2022.
- [70] Kai-Yun Tsao, Thomas Girdler, and Vassilios G. Vassilakis. A survey of cyber security threats and solutions for uav communications and flying ad-hoc networks. *Ad Hoc Networks*, 133:102894, 2022.

- [71] Kai-Yun Tsao, Thomas Girdler, and Vassilios G. Vassilakis. A survey of cyber security threats and solutions for UAV communications and flying ad-hoc networks. Ad Hoc Networks, 133:102894, 2022.
- [72] H. Wang, G. Ding, J. Chen, Y. Zou, and F. Gao. UAV anti-jamming communications with power and mobility control. IEEE Transactions on Wireless Communications, 22(7):4729–4744, July 2023.
- [73] X. Wang, Z. Zhao, L. Yi, Z. Ning, L. Guo, F. R. Yu, and S. Guo. A survey on security of UAV swarm networks: Attacks and countermeasures. ACM Computing Surveys, 57(3), 2024.
- [74] Jean-Paul Yaacoub, Hassan Noura, Ola Salman, and Ali Chehab. Security analysis of drones systems: Attacks, limitations, and recommendations. Internet of Things, 11:100218, 2020.
- [75] Jean-Paul Yaacoub, Hassan Noura, Ola Salman, and Ali Chehab. Security analysis of drones systems: Attacks, limitations, and recommendations. Internet of Things, 11:100218, 2020.
- [76] Jean-Paul A. Yaacoub, Hassan N. Noura, Ola Salman, and Ali Chehab. Robotics cyber security: Vulnerabilities, attacks, countermeasures, and recommendations. International Journal of Information Security, 21:115–158, 2022.
- [77] Jingpu Yang, Mingxuan Cui, Hang Zhang, Fengxian Ji, Zhengzhao Lai, and Yufeng Wang. Agent-based anti-jamming techniques for UAV communications in adversarial environments: A comprehensive survey, 2025.

Keep in touch



[PiQASO LinkedIn](#)



[PiQASO Facebook](#)



[PiQASO Youtube](#)



[ECCC Newsletter](#)



[ECCC LinkedIn](#)



[ECCC Twitter/X](#)



[ECCC Website](#)