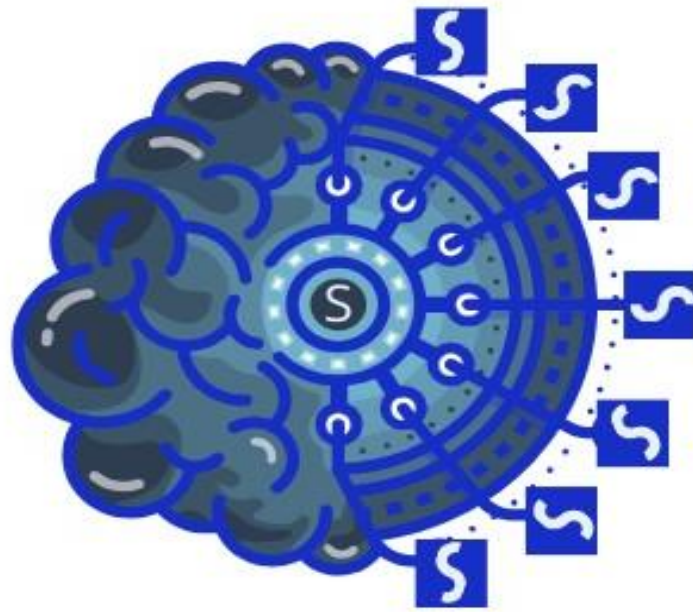


**SCALABLE AND QUANTUM RESILIENT  
HETEROGENEOUS EDGE COMPUTING  
ENABLING TRUSTWORTHY AI**



**SMARTY**



*This project is supported by the European Union's HORIZON-JU-IA under grant agreement No. 101140087*

# D3.1 First report on hardware accelerators for the secure communications

---

---

<b>Editor</b>	Borja Saez (IFAG), Antonio Escobar (IFAG)
<b>Contributors</b>	IFAG, CNIT, CEF, TUE, UOS, BSC, UPM, NVIL, ORL, HTEC, TUB, BOSCH, NEXE
<b>Version</b>	1.0
<b>Date</b>	05, 23, 2025
<b>Distribution</b>	PUBLIC (PU), SENSITIVE (SEN), DMP

**DISCLAIMER**

This document contains information which is proprietary to the SMARTY (Scalable and Quantum Resilient Heterogeneous Edge Computing enabling Trustworthy AI) consortium members that is subject to the rights and obligations and to the terms and conditions applicable to the Grant Agreement number 101140087. The action of the SMARTY consortium members is funded by the European Commission.

Neither this document nor the information contained herein shall be used, copied, duplicated, reproduced, modified, or communicated by any means to any third party, in whole or in parts, except with prior written consent of the SMARTY consortium members. In such a case, an acknowledgement of the authors of the document and all applicable portions of the copyright notice must be clearly referenced. In the event of infringement, the consortium members reserve the right to take any legal action they deem appropriate.

This document reflects only the authors' view and does not necessarily reflect the view of the European Commission. Neither the SMARTY consortium members as a whole, nor a certain SMARTY consortium member warrant that the information contained in this document is suitable for use, nor that the use of the information is accurate or free from risk and accepts no liability for loss or damage suffered by any person using this information.

The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

## REVISION HISTORY

Revision	Date	Responsible	Comment
0.1	April, 1, 2025	IFAG	ToC
0.2	April, 30, 2025	WP3 partners	Initial version
0.3	May, 10, 2025	CNIT	Quality review
0.4	May, 20, 2025	IFAG	Final

## LIST OF AUTHORS

Partner	Name Surname
IFAG	Antonio Escobar, Borja Sáez
CNIT	Filippo Cugini, Michelangelo Guaitolini
CEF	Valentina Carriero, Marco Grassi, Alessio Carenini
TUE	Bruno Cimoli
UOS	Cosimo Gregucci, Jiaxin Pan, Arvinth Arunbabu.
BSC	Jeremy Jens Giesen, Sergi Alcaide
UPM	Javier Conde, Alvaro Alonso, Gabriel Huecas
HTEC	Adrian Garcia Jimenez, Uyen Nguyen
NVIL	Juan Jose Vegas Olmos
TUB	Danh Le Phouc

## EXECUTIVE SUMMARY

The “D3.1 First Report on Hardware Accelerators for Secure Communications” outlines critical advancements in developing hardware-based solutions for secure communication systems. The report focuses on the Quantum-Resistant Secure Element (QR-SE) and Quantum Resistant (QR) transceiver, designed to address emerging threats such as quantum computing, and highlights key achievements, including requirement analysis and the integration of post-quantum cryptographic algorithms like KYBER and DILITHIUM. Additionally, it emphasizes the use of the Linked Open Terms (LOT) methodology to ensure a structured, collaborative, and future-proof development process. By combining innovative hardware accelerators with ontology-driven design, the project aims to deliver scalable, high-performance, and secure communication solutions that meet evolving technological and security demands.

---

## TABLE OF CONTENTS

---

1	INTRODUCTION.....	5
2	Development of a Quantum-Resistant Secure Element .....	6
2.1	Quantum-Resistant Secure Element.....	6
2.2	Ultra-Low Power Processor for PQC and Edge-AI.....	8
2.2.1	Hardware Development.....	8
2.2.2	Software Development .....	8
2.2.3	Demonstrator .....	9
2.2.4	Next Steps.....	11
3	Development of a Quantum-Resistant Transceiver .....	11
3.1	General Overview .....	11
3.2	System architecture and parameters .....	11
3.3	W-band prototype .....	12
3.4	Next steps.....	14
4	PQC Acceleration for Edge Computing Infrastructure .....	14
4.1	General Overview .....	14
4.2	Purpose and design approach .....	14
4.3	Integration into the system architecture.....	15
4.4	Module Breakdown and Pipelined Execution .....	15
4.5	Interfaces and Configuration Registers.....	18
4.5.1	ML-KEM AXI-Lite Control Interface .....	18
4.5.2	ML-KEM AXI-Full Data Interface.....	18
4.5.3	ML-DSA AXI-Lite Control Interface.....	19
4.5.4	ML-DSA AXI-Full Data Interface .....	19
5	SafeSU.....	20
5.1	General Information .....	20
5.2	Purpose and Scope .....	20
5.3	Place in the Evaluation System .....	21
5.4	Block Diagram.....	21
5.5	Interfaces.....	22
5.5.1	AMBA AHB/AXI Interfaces.....	22
5.5.2	AMBA APB interface .....	22
6	Semantic Management of Things and Embedded AI and NLP .....	28
6.1	Support to interoperability along the stack .....	28

---

6.1.1	Interoperability-oriented metadata ontology.....	29
6.1.2	Service mediation through declarative semantic-web based mappings.....	35
6.2	Embedded AI and NLP .....	37
6.3	Hardware/Software integration.....	38
6.3.1	Semantic Management of Quantum-Resistant Secure Element (Asset #2).....	38
6.3.2	Semantic Management of Ultra-low Power Processor for PQC and Edge-AI (Assets #1 & #24) .....	39
6.3.3	Semantic Management of Programmable Data Plane Testbed (Asset #10) .....	41
6.3.4	Semantic Management of Post Quantum Computing Accelerator (Asset #4) .....	42
6.3.5	Semantic Management of Safe Statistics Unit (SafeSU) (Asset #6).....	42
7	CONCLUSIONS .....	42
8	REFERENCES .....	43

## TABLE OF FIGURES

---

Figure 1: Secure Element JavaCard OS Architecture with PQC Applets.....	7
Figure 2: Ultra-low power processor for PQC and Edge-AI Basic Demo setup .....	9
Figure 3: Basic Demo GUI.....	10
Figure 4: Ultra-Low power processor for PQC and edge-AI Basic Demo Sequence Diagram .....	10
Figure 5 HLS-PQC - Place in the system.....	15
Figure 6 HLS-PQC - ML-DSA Sign .....	16
Figure 7 HLS-PQC - ML-KEM Encapsulation (left) and Decapsulation (right).....	17
Figure 8 SafeSU - Place in the evaluation system .....	21
Figure 9 SafeSU - Block diagram.....	21
Figure 10 SafeSU - Base Configuration Register (0x000) .....	22
Figure 11 SafeSU - Crossbar Configuration Register 0 (0x0AC).....	22
Figure 12 SafeSU - Crossbar Configuration Register 1 (0x0B0) .....	23
Figure 13 SafeSU - Crossbar Configuration Register 2 (0x0B4) .....	23
Figure 14 SafeSU - Crossbar Configuration Register 3 (0x0B8) .....	23
Figure 15 SafeSU - Overflow Interrupt Enable Mask (0x064) .....	24
Figure 16 SafeSU - Overflow Interrupt Vector (0x068) .....	25
Figure 17 SafeSU - Block diagram of the MCCU mechanism for one core.....	25
Figure 18 SafeSU - MCCU Main Configuration (0x074) .....	25
Figure 19 SafeSU - MCCU Event Weights Register 0 (shared with RDC; 0x098) .....	26
Figure 20 SafeSU - MCCU Event Weights Register 1 (shared with RDC; 0x09c) .....	26
Figure 21 SafeSU - Block diagram of the RDC mechanism .....	27
Figure 22 SafeSU - RDC Interrupt Vector (0x0A0) .....	27
Figure 23 SafeSU - RDC Event Weights Registers 0 and 1 (shared with MCCU; 0x098, 0x09C)..	28
Figure 24 SafeSU - RDC Watermark Registers 0 and 1 (0x0A4, 0x0A8) .....	28
Figure 25. LOT methodology for ontology development.....	30
Figure 26. Overview of requirements collected w.r.t types of assets .....	31
Figure 27. Example of requirements collected w.r.t. relevant metadata of assets.....	32
Figure 28: Any-to-any versus any-to-one mapping approaches.....	35
Figure 29: Lifting and lowering steps for the RDF Any-to-One Mapping Approach .....	36

## TABLE OF TABLES

---

Table 1: Different key formats for ML-KEM and ML-DSA .....	7
Table 2: Specifications QR transceiver. ....	12
Table 3 Configuration registers managed by s_axi_control (AXI-Lite Interface) .....	18
Table 4 HLS-PQC - AXI-Full Data Interface.....	19
Table 5 HLS-PQC - Configuration registers managed by s_axi_control (AXI-Lite Interface) .....	19
Table 6 HLS-PQC - AXI-Full Data Interface.....	20
Table 7 Crossbar outputs and SafeSU capabilities .....	24

## TABLE OF ABBREVIATIONS AND ACRONYMS

ADMS	Asset Description Metadata Schema
AES	Advanced Encryption Standard
AGV	Automated Guided Vehicles
AHB	Advanced High-performance Bus
AI	Artificial Intelligence
AIA	AI Act
AIoT	Artificial Intelligence of Things
ALTAI	Assessment List for Trustworthy AI
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
AR	Augmented Reality
AXI	Advanced eXtensible Interface
BLE	Bluetooth Low Energy
BPSK	Binary Phase Shift Keying
CCS	Contention-Cycle Stack
DCAT	Data Catalog Vocabulary
DGA	Data Governance Act
DPU	Data Processing Unit
ED	Envelope Detector
EGTAI	Ethics Guidelines for Trustworthy AI
EMDC	Edge Micro Data Center
EU	European Union
FPGA	Field-Programmable Gate Array
FRIA	Fundamental Rights Impact Assessment
GDPR	General Data Protection Regulation
GNN	Graph Neural Network
GPU	Graphics Processing Unit
HIC	Human-in-command
HITL	Human-in-the-loop
HLEG	High-Level Expert Group
HLS	High-Level Synthesis
HOTL	Human-on-the-loop
IoT	Internet of Things
ITSMO	IT Service Management Ontology
JSON	JavaScript Object Notation
KPI	Key Performance Indicators
KPIs	Key Performance Indicators
LNA	Low Noise Amplifier
LOT	Linked Open Terms
MCCU	Maximum-Contention Control Unit
MDR	Medical Device Regulation

ML	Machine Learning
ML-DSA	Machine Learning - Digital Signature Algorithm
ML-KEM	Machine Learning - Key Encapsulation Mechanism
MPA	Medium Power Amplifier
MQTT	Message Queuing Telemetry Transport
MTL	Mapping Template Language
MZM	Mach-Zehnder Modulator
NGO	Non-Governmental Organization
NIC	Network Interface Card
NLP	Natural Language Processing
OBO	Open Biological and Biomedical Ontology Foundry
OECD	Organization for Economic Co-operation and Development
OOK	On-Off Keying
OWL	Web Ontology Language
PAM	Pulse Amplitude Modulation
PQC	Post-Quantum Cryptography
PROV-O	Provenance Ontology
QPSK	Quadrature Phase Shift Keying
QR-SE	Quantum-Resistant Secure Element
QSFP	Quad Small Form-factor Pluggable
RDC	Request Duration Counter
RDF	Resource Description Framework
RDF4J	RDF for Java
RML	RDF Mapping Language
RNG	Random Number Generator
SBOM	Software Bill of Materials
SFP	Small Form-factor Pluggable
SHACL	Shapes Constraint Language
SoC	System on Chip
SOSA	Sensor, Observation, Sample, and Actuator ontology
SPDX	Software Package Data Exchange
SSN	Semantic Sensor Network ontology
SWO	Software Ontology
UTC-PD	Uni-Traveling Carrier Photodiode
VSS	Vehicle Signal Specification
XAI	Explainable AI
YAML	Yet Another Markup Language

# 1 INTRODUCTION

---

The D3.1 First Report on Hardware Accelerators for Secure Communications aimed at addressing the growing need for secure and efficient communication frameworks. With the rising complexity of interconnected systems, such as IoT devices, edge computing, and AI-based applications, the report focuses on leveraging hardware accelerators to meet the dual challenges of performance and security. These accelerators are purpose-built to optimize cryptographic functions, offering enhanced computational efficiency and resistance to emerging cyber threats, including those posed by quantum computing.

A key focus of the report is the development of a Quantum-Resistant Secure Element (QR-SE), designed to future-proof communication systems against the anticipated capabilities of quantum computers. Through comprehensive requirement analysis, critical hardware and software needs have been identified, including memory allocations for cryptographic operations, ensuring compatibility with upcoming cryptographic standards. Furthermore, the project aims to develop a quantum-resistant (QR) transceiver prototype for high-speed, secure wireless communication in the W-band (75-110 GHz). With a target data rate of up to 10 Gbps and AES-256 encryption, it ensures efficient and secure connections in fiberless environments.

Another notable aspect of the report is the emphasis on ontology-driven development. The project adopts the Linked Open Terms (LOT) methodology to create a structured framework for hardware and software design, ensuring seamless integration into complex systems. This approach is highly collaborative and iterative, involving workshops and ongoing stakeholder engagement to align development efforts with real-world use cases. By defining requirements, functionality, and vocabulary early in the process, the SMARTY team is minimizing ambiguities and creating solutions that are transparent, scalable, and interoperable.

The report is structured to provide insights into the challenges, progress, and methodologies implemented in the early stages of the project. It not only outlines technical advancements but also serves as a foundational blueprint for subsequent deliverables. With its focus on innovation and practicality, this document addresses both immediate needs and long-term objectives in securing communication systems.

## 2 DEVELOPMENT OF A QUANTUM-RESISTANT SECURE ELEMENT

---

### 2.1 QUANTUM-RESISTANT SECURE ELEMENT

The progress report for the Quantum-Resistant Secure Element (QR-SE) in the SMARTY project highlights significant advancements across several activities, focusing on the development of key components for secure and efficient cryptographic functionality. The primary objective is to design a secure element with post-quantum cryptography (PQC) capabilities, ensuring compatibility with future cryptographic standards while addressing performance and resource constraints.

Work on requirement analysis for the secure element with PQC functionality has been successfully completed. This analysis defined critical hardware and software needs, including specific memory allocations for cryptographic operations (e.g., 30 KB for key encapsulation mechanisms and 30 KB for digital signature algorithms), the use of an ARM v8-M architecture core, integrated hardware accelerators for PQC, and performance benchmarks (e.g., 100ms for key establishment and 500ms for signature generation). These requirements serve as the foundation for subsequent development tasks and will be refined as necessary during implementation.

Building on this, the conceptual architecture of the secure element was finalized. This concept ensures support for multiple communication interfaces, secure data transmission, and an optimized heap structure for better performance during data access and storage. The architecture incorporates Global Platform Specification 2.3.1 for application management, a Java Card Virtual Machine for platform security, and a Java Card Firewall for applet isolation and memory access control. These design elements provide a robust framework to develop and test the OS in subsequent stages.

The development of the security controller and its OS based on the finalized concept is currently underway. Hardware adjustments are being made to accommodate the increased memory and performance demands of PQC procedures while optimizing RAM usage to prevent unnecessary hardware expansions. Simultaneously, the OS is being designed to handle various communication protocols securely, manage heap memory efficiently, and implement Java Card APIs compliant with Java Card Specification 3.2. Proprietary APIs are also under development to support PQC algorithms and are being designed in alignment with future Java Card standardization efforts. These APIs aim to balance memory optimization and high performance during cryptographic operations.

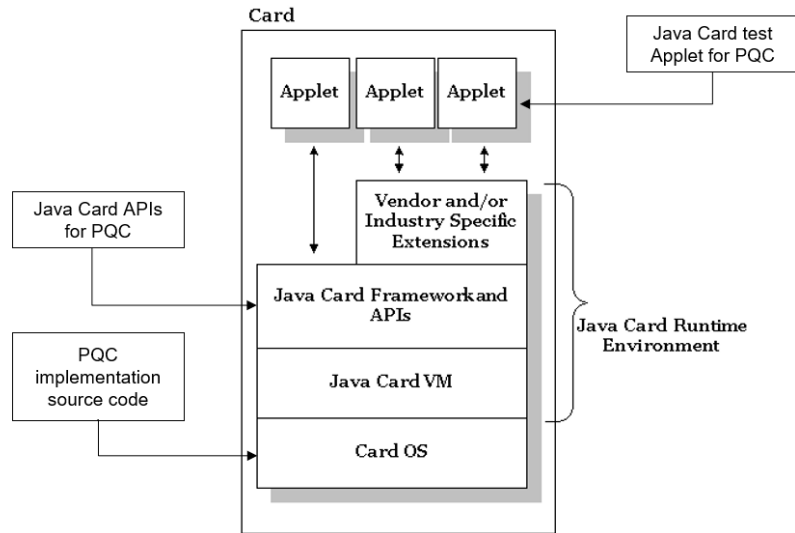


Figure 1: Secure Element JavaCard OS Architecture with PQC Applets

Progress has also been made in the development of the PQC kernel library for key encapsulation. Core routines, such as ring arithmetic, number-theoretic transforms (NTT), and sampling, have been implemented with hardware accelerators, significantly improving performance and reducing code size. A conceptual design for scheme-layer routines, including key generation and key encapsulation/decapsulation, has been completed, and a self-testing application note has been created to validate the library's basic functionalities. The next phase will focus on refining these routines, implementing additional features, and ensuring compliance with previously established requirements.

Table 1: Different key formats for ML-KEM and ML-DSA

Approach	Key size in bytes (public key / secret key)	Comment
ML-KEM original	800 to 1568 / 1632 to 3168	Largest size but fastest solution
ML-DSA original	1312 to 2592 / 2560 to 4896	Secret key is already compressed
ML-KEM seed-based key generation	2 x 32	Smallest variant but slowest
ML-DSA seed-based key generation	32	Smallest variant but slowest
ML-KEM optimized key representation	800 to 1568 / 1056 to 2016	Compromise between size and performance
ML-DSA optimized key representation	1312 to 2592 / 2560 to 4896	Secret key is already compressed

In the area of key formats, the finalized concept introduce two approaches to balance key size and runtime performance for PQC algorithms. The first approach uses seed-based key generation for minimal key storage but incurs increased runtime costs due to frequent recomputation. The second approach, optimized key representation, offers a trade-off by reducing key size (reduction in security) while maintaining better performance. These concepts will be integrated into the key encapsulation mechanism (KEM) library to ensure efficient and secure key management.

Overall, the SMARTY project has made meaningful strides toward realizing a secure and efficient QR-SE with robust PQC capabilities. The completed requirement analysis and architectural design provide a strong foundation for ongoing development, while advancements in hardware, software, and cryptographic libraries demonstrate steady progress toward achieving project goals. Next steps include continued development of OS features, refinement of PQC kernel libraries, and integration of proprietary APIs to ensure a seamless implementation of PQC functionality in the chip card ecosystem.

While significant progress has been made, the SMARTY project continues to address remaining challenges in the development of its QR-SE. Tasks such as revising hardware co-processors for PQC, integrating side-channel countermeasures, and preparing the OS for secure in-field updates are yet to begin. Furthermore, the full integration of the developed PQC kernel library into the Java Card OS and the implementation of hybrid key encapsulation procedures remain essential for achieving the project's overarching goals. These efforts will be reflected in D3.2 and D3.3.

## 2.2 ULTRA-LOW POWER PROCESSOR FOR PQC AND EDGE-AI

Since the last reporting period, significant progress has been made in the development of the *Ultra-Low Power Processor for PQC and Edge-AI*, covering both hardware and software components, as well as the design of an initial demonstrator.

### 2.2.1 Hardware Development

During this phase, while the final QR-SE (Secure Element) is still under development, an alternative Secure Element provided by IFAG has been utilized. This interim SE operates using the same Smart Card OS and supports I2C communication, making it suitable for ongoing development activities.

In parallel, a PSoC Edge Development Kit, a platform including processor and SE, has been employed as a temporary hardware platform until the final design is ready for manufacturing. This provisional SE does not include the IEC7816 interface, which will be integrated once the final hardware becomes available. Despite this limitation, the current setup supports the majority of development tasks. Once the final SE is deployed, the existing cryptographic algorithms will be replaced with post-quantum cryptography (PQC) counterparts, and the IEC7816 interface will be incorporated.

To enable direct communication with the Secure Element from a PC, an I2C-to-USB interface device (Aardvark) has been used. This device includes drivers and libraries compatible with multiple programming languages, allowing for easy integration across various platforms. Additionally, IFAG provides example implementations that demonstrate how to use the SE with the Aardvark device.

Finally, to support remote development and continuous integration workflows, the Aardvark device has been connected to a development server, enabling remote access to SE functionalities.

### 2.2.2 Software Development

The initial step in the software development for the Ultra-Low Power Processor for PQC and Edge-AI involved testing the basic example provided with the Secora Connect Secure Element

integrated within the PSoC Edge Development Kit from IFAG. This example application runs on a Windows environment and uses the Aardvark interface to establish communication with the SE. It demonstrates how to enable an applet and configure the NFC interface.

In addition to the example code, Infineon also provides a standar Smart Card Stack implementation. For its use in SMARTY, this stack was adapted for use with the PSoC Edge platform by integrating the necessary hardware drivers. Once the adaptation was completed and verified, the original SE example was successfully reproduced on the PSoC Edge hardware, eliminating the need for a host computer in the loop.

To support remote development and continuous integration, *gRPC* was integrated into the project. This allows a remote service to expose the Aardvark driver APIs, enabling access to the SE over the network. The corresponding client API can be integrated into test and demonstration software. This setup enables automated testing with the physical device upon every update pushed to the repository, streamlining the development and validation process.

### 2.2.3 Demonstrator

To support testing and validation, a basic demonstrator has been developed. Since the final PQC algorithms are not available yet, an applet implementing AES-based cryptography was installed on the Secora Connect Secure Element (SE) so the system can be tested.

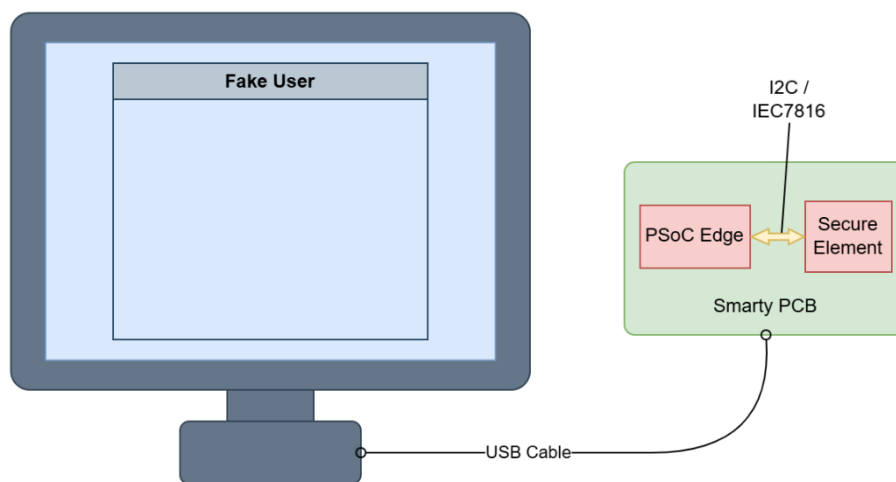


Figure 2: Ultra-low power processor for PQC and Edge-AI Basic Demo setup

In this demonstration setup, the PSoC Edge board is connected to a PC via USB/COM, while the SE communicates with the PSoC Edge through the I2C interface. A Windows application, referred to as "Fake User," is responsible for sending data and corresponding digital signatures to the PSoC Edge. These signatures can either be valid or deliberately invalid to simulate different scenarios.

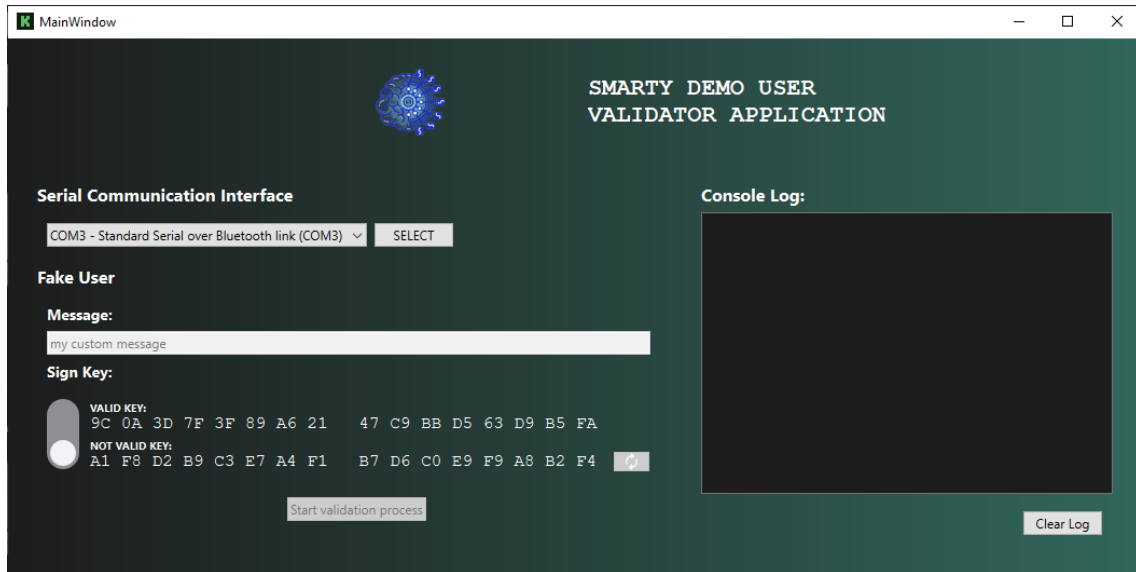


Figure 3: Basic Demo GUI

Upon receiving a message, the PSoC Edge processes the data and forwards it to the SE to generate a new signature using the shared key. It then compares the signature received from the Fake User with the one generated by the SE. Based on this comparison, it sends a response back to the Fake User application indicating whether the signature was valid.

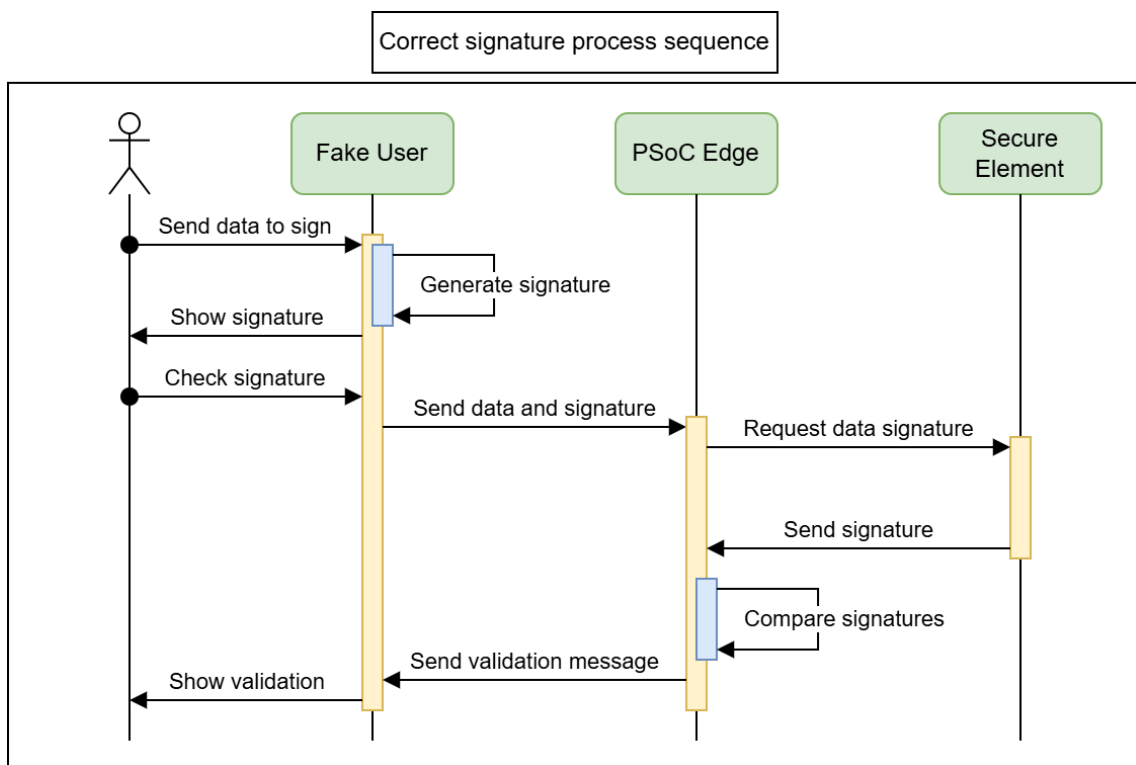


Figure 4: Ultra-Low power processor for PQC and edge-AI Basic Demo Sequence Diagram

To ensure modularity and ease of reuse, the applet functionality and SE communication logic have been encapsulated into a dedicated class. This abstraction facilitates per integration into other applications or future extensions of the current demonstration.

2.2.4 Next Steps

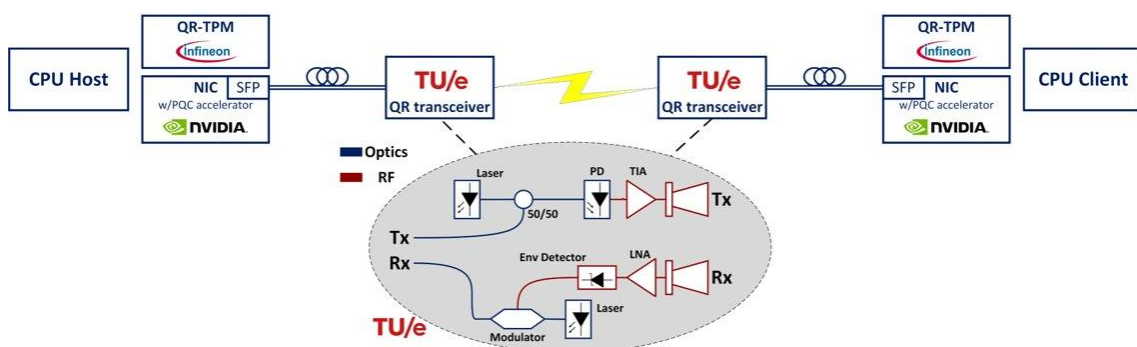
The following steps are planned to advance the development of the Ultra-Low Power Processor for PQC and Edge-AI:

- Test the Final Secure Element (SE): Validate the final version of the SE in the target hardware environment.
- Integrate Post-Quantum Algorithms: Replace the placeholder AES algorithm with the intended post-quantum cryptographic (PQC) algorithms and ensure full functional integration.
- Adapt the Demonstrator: Update the basic demonstration application to work seamlessly with the final SE, incorporating all new cryptographic features.
- Define Final Hardware Interfaces: Specify and design the required hardware interfaces for integration with target use cases, including 5G, CAN bus, and other relevant technologies.
- Finalize Hardware Design: Complete the hardware design phase and initiate manufacturing of the final device.

### 3 DEVELOPMENT OF A QUANTUM-RESISTANT TRANSCEIVER

#### 3.1 GENERAL OVERVIEW

TUE will develop a prototype of a quantum resistant (QR) transceiver for short range and/or chip-to-chip high speed wireless communications over the W-band (75-110 GHz). The aim is to develop a high speed, low latency, low complexity and quantum secure wireless connection between edge units that are temporarily deployed on the field when no direct fiber connection is available.



#### 3.2 SYSTEM ARCHITECTURE AND PARAMETERS

The following table summarizes the main parameters for the QR transceiver. These transmitters have been defined in WP2. Two possible transceivers will be tested for W- and J-band. For the prototype the W-band system will be implemented with a target maximum data rate of 10 Gbps for OOK and 4-PAM modulations. Phase modulations schemes will be investigated, although it would require coherent detection and low phase noise sources.

Table 2: Specifications QR transceiver.

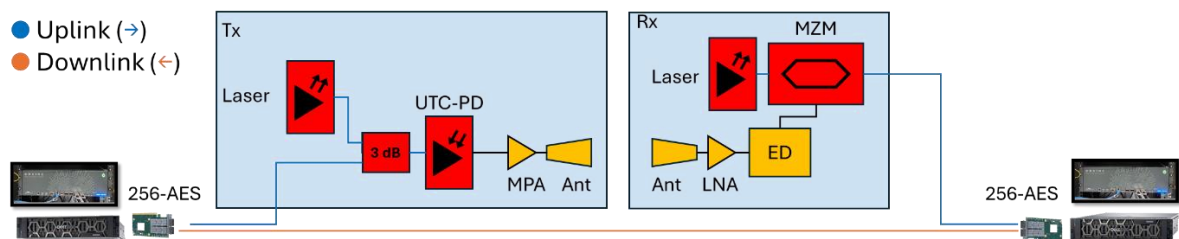
<b>Frequency Band (GHz)</b>	W-band (75-110), J-band (220-330)
<b>Max data rate (Gbps)</b>	10 (W-band), 100 (W-band)
<b>Modulation formats</b>	OOK, PAM, QPSK, BPSK
<b>Wireless Transmitter</b>	UTC Photodiode
<b>Wireless Receiver</b>	Envelope Detector/Mixer
<b>Photonic transceivers</b>	SFP, SPF+, QSFP
<b>Multiplexing</b>	Frequency (same as SFPs in fiber)
<b>Distance (m)</b>	5-10
<b>Security element</b>	Smart NIC
<b>Encryption</b>	256-AES

### 3.3 W-BAND PROTOTYPE

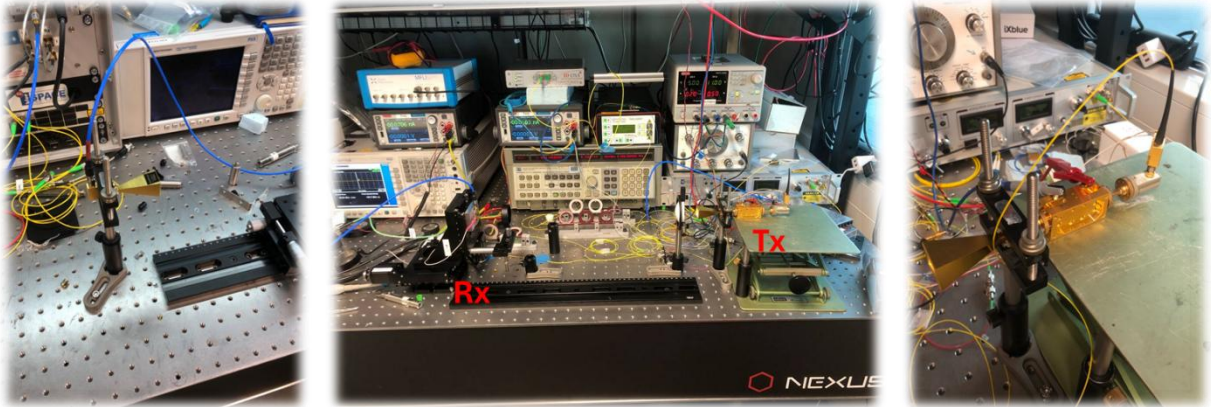
The first step for building the QR prototype is the laboratory setup of the bidirectional W-band link. The current state of the setup is shown in the following schematic. We can observe the wireless link is unidirectional, only the uplink (UL), while the downlink (DL) is still only on fiber.

At the transmitter side, we have a coherent detection receiver in the fiber followed by a W-band transmitter. In the fiber the incoming data signal is combined with another laser, which acts as a local oscillator (LO); the two tones are then fed to the uni-traveling-carrier photodiode (UTC-PD) that generates the WQ-band signal by beating the two laser tones. The UTC-PD is fast enough to output signals in the W-band. The wireless transmitter consists of a single medium power amplifier (MPA) and a horn antenna.

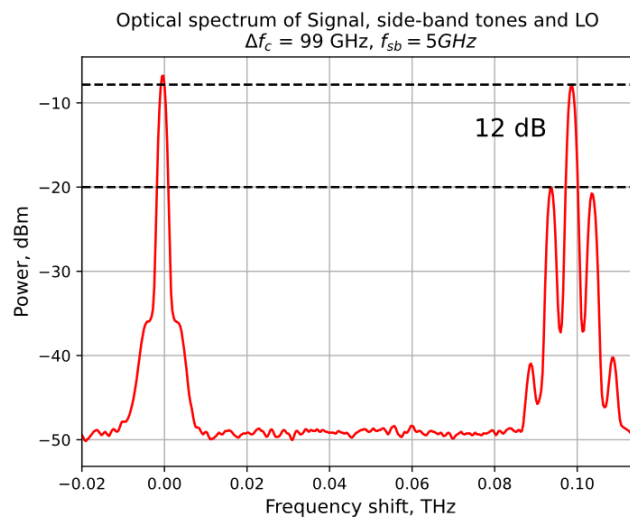
The Wireless receiver consists of a horn antenna, a low noise amplifier (LNA) and an envelope detector (ED), which performs the amplitude demodulation via direct detection. The output of the ED is a baseband signal that is fed to a Mach-Zender modulator (MZM), which convert back to the optical domain if driven by a laser as depicted in the following figure.



Currently laboratory test is done with random data generated by a signal generator and an oscilloscope as a receiver to record the output signal. We predict to switch to the SFP transceivers within a few months. The following pictures show the laboratory setup. In this configuration no LNA was applied so the distance was reduced to 0.5 meters. We will increase the distance to 5-10 meters by applying the LNA after the down-link set up.



The following figure shows the optical spectrum at the input of the photodiode. We can notice the two optical signals: data and laser LO. For this measurement the data consisted of an intermediate carrier of 5 GHz.



The following figure shows the eye diagram of a 1.3 Gbps OOK signal transmitted over the wireless link with a carrier frequency of 80 GHz. The measurement point is at the output of the envelope detector. We can notice from the eye opening that the signal is error free.



### 3.4 NEXT STEPS

The next step is to upgrade the current setup to be bidirectional. This will require multiplexing the two links. We decided to apply frequency multiplexing as it is commonly done in mobile networks as well as SFP modules in the fiber. More specifically the UL will be placed on the lower half of the W-band (75-92 GHz) and the DL on the higher half (93-110 GHz). Two possible schematics will be considered for the multiplexing: 4 antennas (a pair per link) or 2 antennas (with circulator). The two antennas approach will be investigated, although the main challenge will be the isolation between transmitter's MPS and receiver's LNA. Other necessary upgrades are the encryptors, phase modulation (after implementing a phase locking scheme for the two lasers) and increase the reach with antenna modules (see following figure).

## 4 PQC ACCELERATION FOR EDGE COMPUTING INFRASTRUCTURE

---

### 4.1 GENERAL OVERVIEW

Ensuring robust communication security in embedded systems has become increasingly critical, especially in light of advances in quantum computing that threaten classical public-key cryptographic algorithms. In response to this evolving landscape, multiple post-quantum cryptographic (PQC) schemes are under evaluation by standardization bodies such as NIST. Among the selected candidates are ML-KEM (FIPS-203) and ML-DSA (FIPS-204), which form the basis for our accelerator implementation.

In this project, BSC will design and integrate a High-Level Synthesis (HLS)-based PQC accelerator into the SELENE SoC, targeting the NOEL-V RISC-V core. The accelerator implements both ML-KEM and ML-DSA cryptographic primitives using a shared modular hardware architecture, allowing for reuse of common components and optimization of hardware resources.

### 4.2 PURPOSE AND DESIGN APPROACH

The goal of this work is to offload cryptographic computations from the general-purpose processor by implementing hardware acceleration for PQC functions. This improves performance, reduces energy consumption, and enhances system responsiveness for security-sensitive operations.

The design approach leverages High-Level Synthesis (HLS) tools, enabling faster prototyping by converting C/C++ descriptions of cryptographic algorithms directly into synthesizable HDL. This methodology accelerates the hardware development cycle and allows for iterative optimization.

Given that ML-KEM and ML-DSA share significant computational structure (e.g., polynomial arithmetic), we will structure the accelerator into modular components that can be reused between the two schemes. These modules are optimized for pipelined and parallel execution to maximize throughput and minimize latency.

### 4.3 INTEGRATION INTO THE SYSTEM ARCHITECTURE

As shown in Figure 2, the accelerator will be connected to the SoC via a Network-on-Chip (NoC) using the AXI4-Full protocol for high-throughput data transfer. Control and configuration are handled via AXI4-Lite, which provides access to memory-mapped registers that manage operational mode selection and I/O pointers.

Each data argument is assigned a dedicated AXI-Full data bus, allowing simultaneous memory transactions during encapsulation, decapsulation, signing, or verification processes. The processor acts as the AXI Master, initiating all control and data transfer operations, while the accelerator behaves as the AXI Slave.

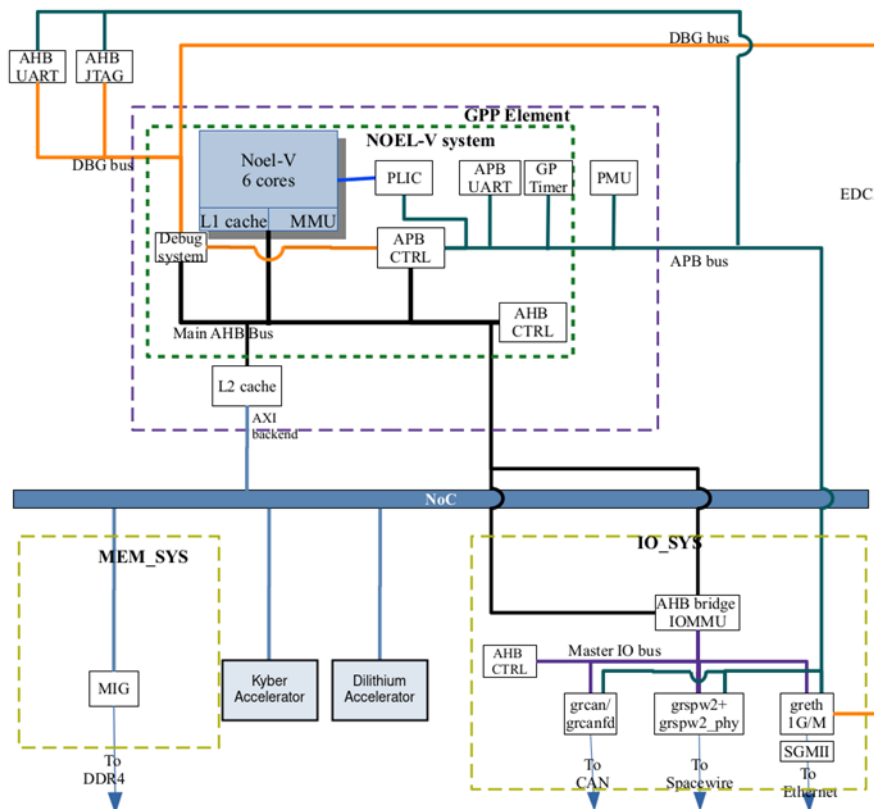


Figure 5 HLS-PQC - Place in the system

This architectural integration ensures compatibility with other SELENE SoC modules and provides the flexibility needed for integration into a variety of edge or embedded platforms.

### 4.4 MODULE BREAKDOWN AND PIPELINED EXECUTION

To achieve optimal performance, each cryptographic function is decomposed into a sequence of independent hardware modules, each responsible for a specific sub-function (e.g., polynomial multiplication, compression, sampling). These are pipelined to allow different stages of the algorithm to be executed in parallel.



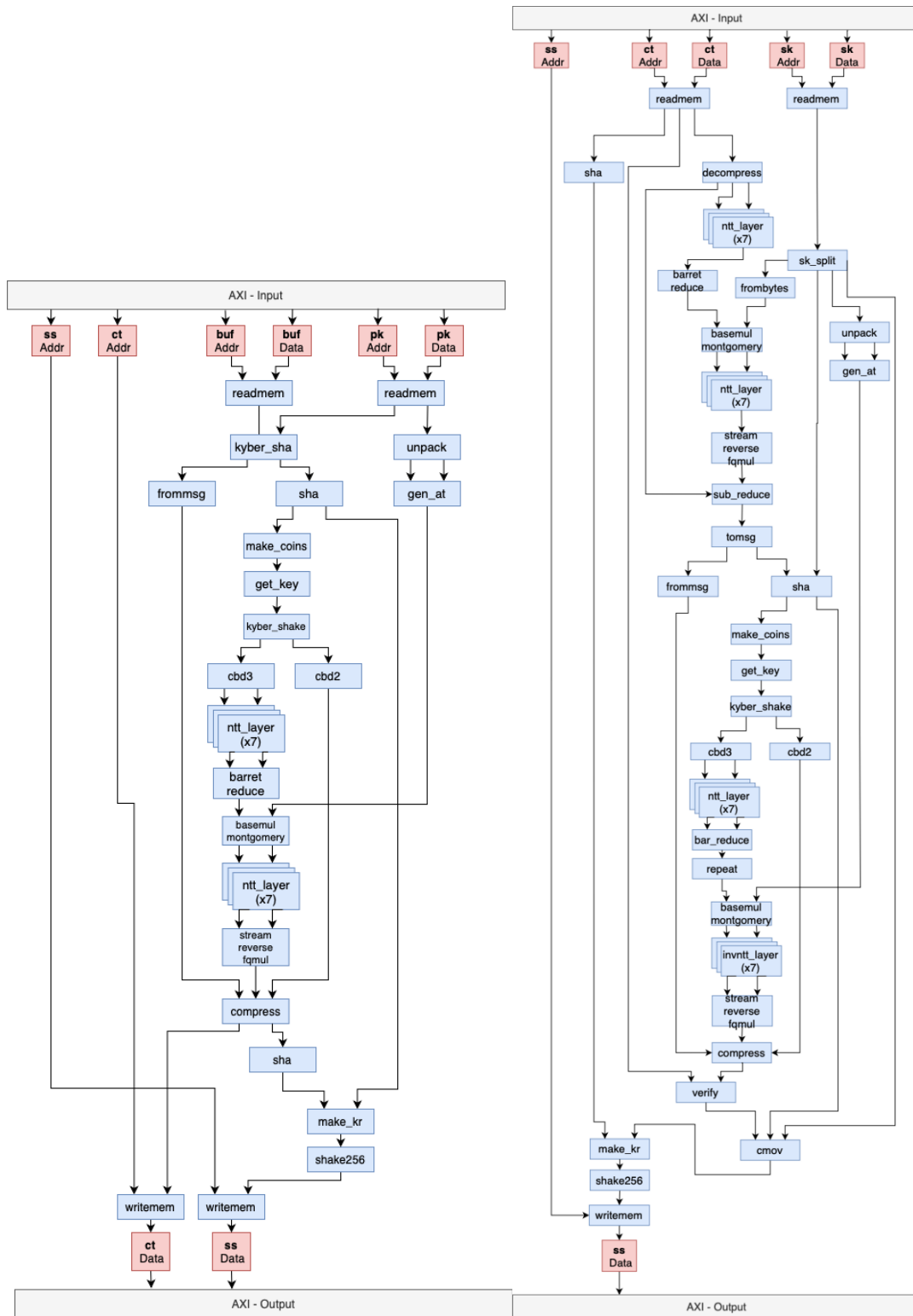


Figure 7 HLS-PQC - ML-KEM Encapsulation (left) and Decapsulation (right)

This decomposition allows shared components (e.g., NTT blocks, random number generators) to be reused across both ML-KEM and ML-DSA, minimizing resource duplication.

## 4.5 INTERFACES AND CONFIGURATION REGISTERS

Each accelerator function is driven by a **memory-mapped register interface** exposed via the AXI-Lite bus. Registers are grouped into:

- Control flags (e.g., start, reset, operation mode)
- Interrupt management (GIER, IER, ISR)
- Memory address pointers for I/O data vectors.

Five **AXI-Full 32-bit data buses** are defined for each scheme, enabling simultaneous access to input and output buffers. The interface protocol ensures synchronization between the processor and accelerator, supporting busy-wait polling and interrupt-based signaling.

### 4.5.1 ML-KEM AXI-Lite Control Interface

The AXI-Lite control interface for the ML-KEM accelerator module manages all configuration, control and interrupt registers. This includes:

- Triggering the start of computation.
- Selecting between encapsulation and decapsulation modes.
- Configuring base addresses for the various input/output data signals.
- Managing interrupt enablement and monitoring status.

The processor writes to these registers to initialize an operation and monitor its status. Bit-level control (e.g., CTRL [0] = 1 to start computation) and explicit memory addressing allow for flexible integration with host software stacks.

Table 3 shows the register map managed via the AXI-Lite interface.

Register	Interface	Offset	Width	Access	Description
CTRL	s_axi_control	0x00	32	RW	Control signals (bit[0]=1 starts computation)
GIER	s_axi_control	0x04	32	RW	Global Interrupt Enable Register
IER	s_axi_control	0x08	32	RW	IP Interrupt Enable Register
ISR	s_axi_control	0x0C	32	RW	IP Interrupt Status Register
kem_cfg	s_axi_control	0x10	64	W	Operation Selection (Enc=0 Dec=1)
ct_in	s_axi_control	0x18	64	W	Memory Address of data signal ct_in
ct.out	s_axi_control	0x20	64	W	Memory Address of data signal ct.out
ss_enc.out	s_axi_control	0x28	64	W	Memory Address of data signal ss_enc.out
ss_dec.out	s_axi_control	0x30	64	W	Memory Address of data signal ss_dec.out
buf_in	s_axi_control	0x38	64	W	Memory Address of data signal buf_in
pk_in	s_axi_control	0x40	64	W	Memory Address of data signal pk_in
sk_in	s_axi_control	0x48	64	W	Memory Address of data signal sk_in

Table 3 Configuration registers managed by s\_axi\_control (AXI-Lite Interface)

### 4.5.2 ML-KEM AXI-Full Data Interface

The AXI-Full interface provides high-bandwidth access to the cryptographic data buffers required during ML-KEM operations. These include the ciphertext, secret and shared keys, and public keys.

Each input/output argument is mapped to a unique AXI addressable region, allowing parallel memory access. The bus width is set to 32 bits to match the memory architecture of the SELENE SoC platform and is optimized for burst transfers.

Table 4 lists the data signals exposed through the AXI-Full interface for the ML-KEM function.

Interface	Argument	Addr Width	Data Width	Access	Description
m_axi_gmembuf	buf	32	32	R	Bit Stream Randomizer Buffer
m_axi_gmemct	ct	32	32	RW	CipherText
m_axi_gmempk	pk	32	32	R	Public Key
m_axi_gmemsk	sk	32	32	R	Secret Key
m_axi_gmemss	ss	32	32	W	Shared Secret

Table 4 HLS-PQC - AXI-Full Data Interface

#### 4.5.3 ML-DSA AXI-Lite Control Interface

The AXI-Lite register interface for the ML-DSA module mirrors the ML-KEM layout in structure but is tailored for digital signature operations. It manages:

- Mode selection between signature generation and verification.
- Start signal activation.
- Address mapping for all relevant I/O buffers (e.g., messages, signatures, keys).
- Interrupt handling.

This uniform design across both ML-KEM and ML-DSA simplifies firmware reuse and reduces the need for interface-specific driver logic.

Table 5 shows the register layout managed through the AXI-Lite interface for ML-DSA.

Register	Interface	Offset	Width	Access	Description
CTRL	s_axi_control	0x00	32	RW	Control signals (bit[0]=1 starts computation)
GIER	s_axi_control	0x04	32	RW	Global Interrupt Enable Register
IER	s_axi_control	0x08	32	RW	IP Interrupt Enable Register
ISR	s_axi_control	0x0C	32	RW	IP Interrupt Status Register
kem_cfg	s_axi_control	0x10	64	W	Operation Selection (Sign=0 Verify=1)
ret_out	s_axi_control	0x18	64	W	Memory Address of data signal ret_out
sign_out	s_axi_control	0x20	64	W	Memory Address of data signal sign_out
sign_in	s_axi_control	0x28	64	W	Memory Address of data signal sign_in
mu_in	s_axi_control	0x30	64	W	Memory Address of data signal mu_in
m_in	s_axi_control	0x38	64	W	Memory Address of data signal m_in
mu2_in	s_axi_control	0x40	64	W	Memory Address of data signal mu2_in
sk_in	s_axi_control	0x44	64	W	Memory Address of data signal sk_in
pk_in	s_axi_control	0x48	64	W	Memory Address of data signal pk_in
ver_out	s_axi_control	0x58	64	W	Memory Address of data signal ver_out
milen_in	s_axi_control	0x60	64	W	Memory Address of data signal milen_in

Table 5 HLS-PQC - Configuration registers managed by s\_axi\_control (AXI-Lite Interface)

#### 4.5.4 ML-DSA AXI-Full Data Interface

Similar to the ML-KEM counterpart, the **AXI-Full interface for ML-DSA** exposes memory-mapped ports for public key input, message input, and signature output buffers. These signals are

organized per task and allow the accelerator to read/write data without blocking the control path.

This separation ensures continuous data streaming across pipeline stages and supports parallelism with other compute units or accelerators sharing the SoC interconnect.

Table 6 defines the data buses exposed through the ML-DSA AXI-Full interface.

Interface	Argument Verify sign	Addr Width	Data Width	Access	Description
m_axi_gmemout	ver_out ret_out	32	32	W	Checks Marking
m_axi_gmemsign	Sign_in Sign_out	32	32	RW	Signature
m_axi_gmemm	mu_orig_In mu_processed_in	32	32	R	Original message Processed message = CRH(H(rho, pk), mu_orig_in)
m_axi_gmempk	Pk_in mu2_processed_In	32	32	R	Public key Processed message = CRH(H(rho, pk), mu_orig_in)
m_axi_gmemsk	sk_in	32	32	R	Secret Key

Table 6 HLS-PQC - AXI-Full Data Interface

## 5 SAFESU

### 5.1 GENERAL INFORMATION

The SafeSU is a modular and scalable Performance Monitor Unit (PMU) that can be connected to any on-chip interconnect and allows multicore interference observability and controllability.

### 5.2 PURPOSE AND SCOPE

The SafeSU builds on a number of components, namely, the Contention-Cycle Stack (CCS), the Request Duration Counter (RDC) and the Maximum-Contention Control Unit (MCCU).

- The CCS offers observability features by providing multicore time-interference breakdown.
- The RDC provides end users with an observability channel to monitor high-watermark latencies per event and core, as needed for interference bounding (e.g., during worst-case execution time estimation).
- The MCCU offers controllability capabilities with interference quota monitoring and enforcement, alerting the user when allocated quotas are exceeded.

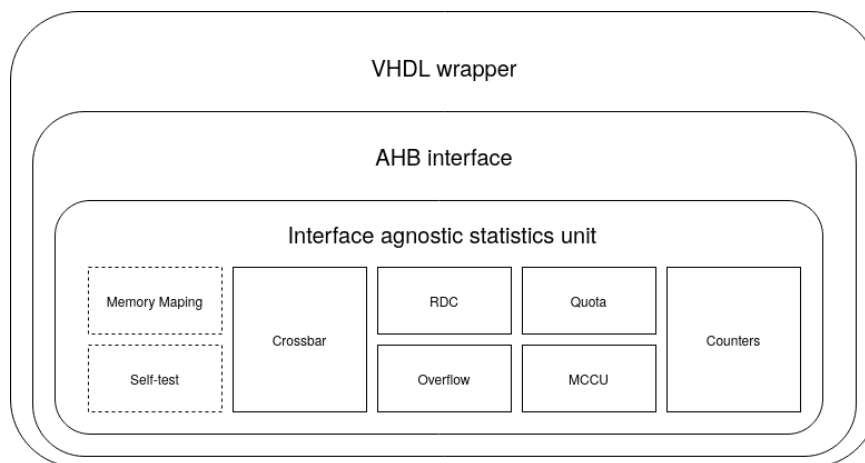
### 5.3 PLACE IN THE EVALUATION SYSTEM

*Figure 8 SafeSU - Place in the evaluation system*

The monitoring interface of the SafeSU depicted in Figure 8, is AMBA AHB and AXI compliant. The SafeSU is intended to be connected to those types of interfaces, and it is particularly useful if those interfaces have either multiple managers or are connected to subordinates receiving requests from multiple managers. For instance, its best location is normally connected to the interface used by the cores and/or accelerators to access shared caches or memory controllers so that ongoing traffic can be monitored, and eventually compared to predefined quotas to ensure that no manager abuses the use of relevant shared resources.

SafeSU's programming port is compliant with AMBA APB, although it will be extended to AMBA AXI in the future.

### 5.4 BLOCK DIAGRAM



*Figure 9 SafeSU - Block diagram*

The main components of the SafeSU are the following:

- **Self-test:** configures the counters' inputs to a fixed value bypassing the crossbar and ignoring the SoC inputs. This mode allows for tests of the software and the unit under known conditions.
- **Crossbar:** routes any input event to any counter.
- **Counters:** A group of simple counters with settable initial values and general control register.
- **Overflow:** Detects counters' overflow. It can raise interruptions upon overflow with its dedicated interruption vector and per counter interrupt enable.
- **Quota:** Deprecated as replaced by MCCU (it may be excluded in a future release).
- **MCCU (Maximum Contention Control Unit):** Contention control measures for each core for a particular event type that has been programmed to be monitored. It can raise an interruption if a contention threshold is exceeded. It accepts real contention signals or estimation through weights.
- **RDC (Request Duration Counters):** Provides measures of the pulse length of a given input signal (watermark). It can be used to determine maximum latency and cycles of

uninterrupted contentions. Each of the counters can trigger an interrupt at a user-defined threshold.

## 5.5 INTERFACES

### 5.5.1 AMBA AHB/AXI Interfaces

The AHB or AXI interface is a subordinate interface used to snoop traffic. It is fully compliant with the specification of the corresponding protocol. Note that, in general, a SafeSU instance supports only one of those interfaces.

### 5.5.2 AMBA APB interface

The AMBA APB subordinate interface is used to program the control registers of the SafeSU. The control registers are as follows:

*Main configuration and self-test*

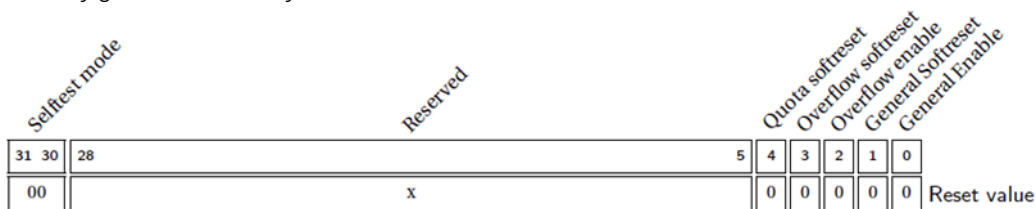


Figure 10 SafeSU - Base Configuration Register (0x000)

Reset and enable of overflow, quota, and regular counters' operations can be performed with the Base Configuration Register shown in Figure 10. All signals are active high.

Self-test mode allows bypassing the input events from the crossbar and instead using a specific input pattern where signals are constant. This mode can be used for debugging. After the addition of the crossbar and debug inputs, there is a certain overlap. The same results can be achieved with the correct crossbar configuration. Nevertheless, it has been included in this release for compatibility.

These are the self-test modes for each configuration value of the field `Selftest mode` part of the register shown in Figure 10:

- 0b00: Events depend on the crossbar. Self-test is disabled.
- 0b01: All signals are set to 1.
- 0b10: All signals are set to 0.
- 0b11: Signal 0 is set to 1. The remaining signals are set to 0.

*Crossbar*

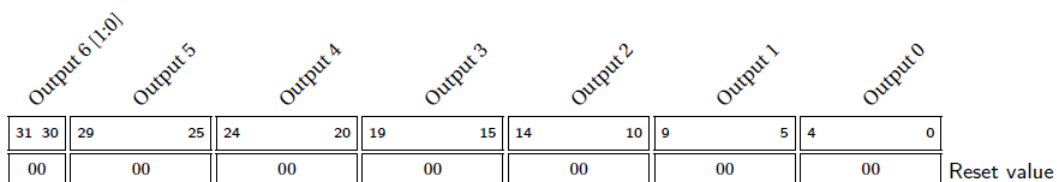


Figure 11 SafeSU - Crossbar Configuration Register 0 (0x0AC)

Output 12(3:0)		Output 11		Output 10		Output 9		Output 8		Output 7		Output 6 (4:2)		
31	28	27	23	22	18	17	13	12	8	7	3	2	0	
00		00		00		00		00		00		00		Reset value

Figure 12 SafeSU - Crossbar Configuration Register 1 (0x0B0)

Output 12(3:0)		Output 11		Output 10		Output 9		Output 8		Output 7		Output 6 (4:2)		
31	28	27	23	22	18	17	13	12	8	7	3	2	0	
00		00		00		00		00		00		00		Reset value

Figure 13 SafeSU - Crossbar Configuration Register 2 (0x0B4)

Reserved		Output 24		Output 23		Output 22		Output 21		Output 20		Output 19(4:1)		
31	29	28	24	23	19	18	14	13	9	8	4	3	0	
x		00		00		00		00		00		00		Reset value

Figure 14 SafeSU - Crossbar Configuration Register 3 (0x0B8)

This feature allows routing any of the input signals of the SafeSU into any of the 24 counters of the SafeSU (see Table 3.3.3.5-1). Each one of the counters has a 5-bit configuration value. These values are stored in the registers shown in Figure 11, Figure 12, Figure 13 and Figure 14. All the configuration values are consecutive. Thus, some values may have configuration bits in two consecutive memory addresses. Examples of this are Output 6, 12, 19 in our current configuration. As a consequence, the previous outputs may require two writes to configure the desired input signal.

Configuration fields match one to one with the internal counters. So, the field Output 0 matches with counter 0, Output 1 with counter 1 and so on.

As a usage example, suppose the user wants to route the signal `pmu_events(0).icnt(0)` to the internal counter 0. The field Output 0 of the register in Figure 11 shall match the index of the signal in the table of inputs. In this case, the index is 2. After this configuration, the event count will be recorded in counter 0. The addresses for counter values range between 0x04 and 0x60.

#### Counters

The unit in the default configuration contains 24 counters, 32-bit each. The memory address where each counter's value can be accessed ranges between 0x04 and 0x60, as said before. Counter values can be read or written, thus allowing to set the initial value of the counters.

Enable and reset are managed by the base configuration register from Figure 10.

Counters can overflow. In such a case, the count will wrap around to 0 and keep counting. The next section (Overflow) describes how to enable the overflow detection interrupts.

Output	Counters	Overflow	MCCU	RDC
0	Yes	Yes	Core 0	Yes
1	Yes	Yes	Core 0	Yes
2	Yes	Yes	Core 1	Yes
3	Yes	Yes	Core 1	Yes
4	Yes	Yes	Core 2	Yes
5	Yes	Yes	Core 2	Yes
6	Yes	Yes	Core 3	Yes
7	Yes	Yes	Core 3	Yes
8	Yes	Yes	No	No
9	Yes	Yes	No	No
10	Yes	Yes	No	No
11	Yes	Yes	No	No
12	Yes	Yes	No	No
13	Yes	Yes	No	No
14	Yes	Yes	No	No
15	Yes	Yes	No	No
16	Yes	Yes	No	No
17	Yes	Yes	No	No
18	Yes	Yes	No	No
19	Yes	Yes	No	No
20	Yes	Yes	No	No
21	Yes	Yes	No	No
22	Yes	Yes	No	No
23	Yes	Yes	No	No

Table 7 Crossbar outputs and SafeSU capabilities

*Overflow*

The user can enable overflow detection for each of the counters in the previous section (Counters). Enables are active high and individual for each counter, as indicated in the Overflow Interrupt Enable Mask register depicted in Figure 15. If a counter with overflow detection active wraps over the maximum value, the corresponding bit of the Overflow Interrupt Vector register depicted in Figure 16 will become 1, and AHB interrupt number 6 will become active.

The default AHB interrupt mapping can be modified within the file *ahb\_wrapper.vhd*.

	Reserved	Counter 23	Counter 22	Counter 21	Counter 20	Counter 19	Counter 18	Counter 17	Counter 16	Counter 15	Counter 14	Counter 13	Counter 12	Counter 11	Counter 10	Counter 9	Counter 8	Counter 7	Counter 6	Counter 5	Counter 4	Counter 3	Counter 2	Counter 1	Counter 0
31	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset value

Figure 15 SafeSU - Overflow Interrupt Enable Mask (0x064)

	Reserved	Counter 23	Counter 22	Counter 21	Counter 20	Counter 19	Counter 18	Counter 17	Counter 16	Counter 15	Counter 14	Counter 13	Counter 12	Counter 11	Counter 10	Counter 9	Counter 8	Counter 7	Counter 6	Counter 5	Counter 4	Counter 3	Counter 2	Counter 1	Counter 0
31	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Reset value

Figure 16 SafeSU - Overflow Interrupt Vector (0x068)

*Quota*

This feature has been replaced by the MCCU and will disappear in future releases. Usage is not recommended.

*MCCU*

The MCCU allows monitoring for a subset of the input events and tracking the approximate contention that they will cause. Currently, events assigned to counters 0 to 7 can be used as inputs of the MCCU. Thanks to the crossbar, any of the 32 SoC signals can be used by the MCCU.

Figure 17 shows the internal elements required to monitor the quota consumption of one core, given that there are four input events. When the events become active, they pass the value assigned in the weight register depicted in Figure 18 for the given signal to a series of adders. The addition is subtracted from the corresponding quota register, mapped to addresses 0x088 to 0x094. If the remaining quota is smaller than the cycle contention, an interruption is triggered.

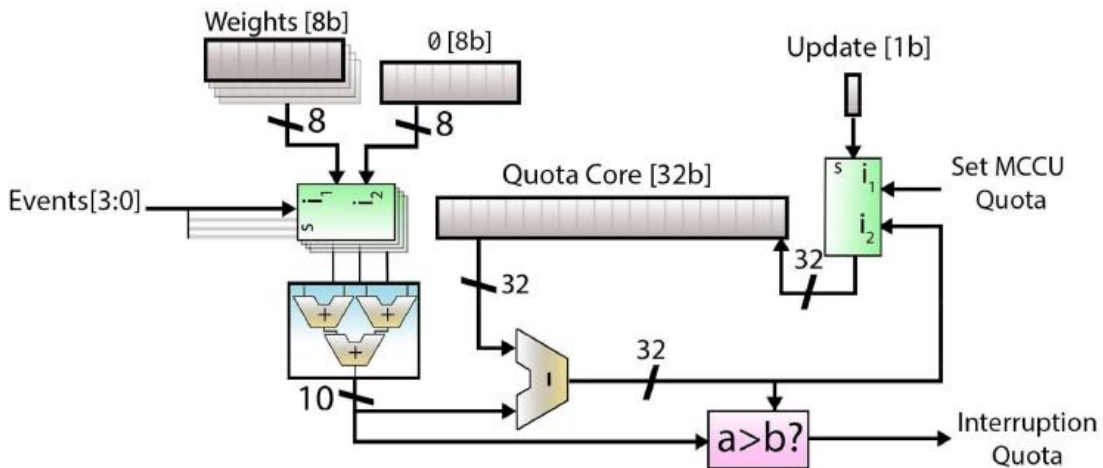


Figure 17 SafeSU - Block diagram of the MCCU mechanism for one core

	Reserved	Soft reset RDC	Enable RDC	Update Quota Core 3	Update Quota Core 2	Update Quota Core 1	Soft reset MCCU	Enable MCCU	
32	8	7	6	5	4	3	2	1	0
	x	0	0	0	0	0	0	0	0

Reset value

Figure 18 SafeSU - MCCU Main Configuration (0x074)

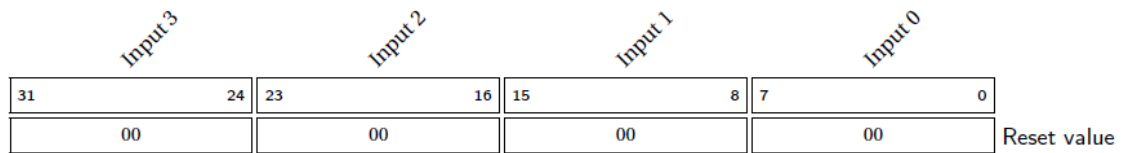


Figure 19 SafeSU - MCCU Event Weights Register 0 (shared with RDC; 0x098)

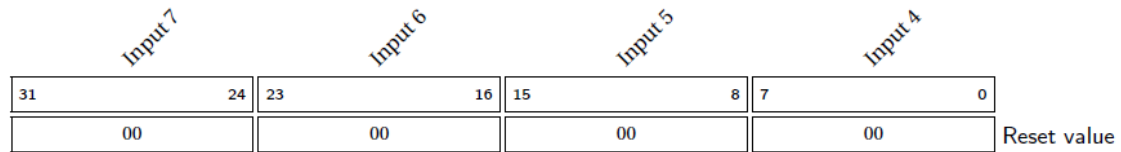


Figure 20 SafeSU - MCCU Event Weights Register 1 (shared with RDC; 0x09c)

In the current release, the MCCU can be reset and activated with the respective fields of the MCCU Main Configuration register depicted in Figure 18. The fields labelled as *Update Quota Core x* are used to update the available quota of each core (addresses 0x088 to 0x094). While *Update Quota Core x* is high, the content of the corresponding quota register (addresses 0x088 to 0x094) is assigned to the available quota, as configured in registers 0x078 to 0x084. Once released (low), the available quota can start to decrease if the MCCU is active. The current quota can be read while the unit is active.

In the current release, each core can monitor two input events. The MCCU module is parametric. More events can be provided in future releases. Table 7 listing the outputs shows the available features for each crossbar output. Under the column MCCU, you can see towards which core quota the event will be computed. The unit provides one interrupt for each of the monitored cores. Quota exhaustion for cores 3, 2, 1, and 0 is mapped to AHB interrupts 10, 9, 8, and 7, respectively.

Weights for each monitored event are registered in the MCCU Event Weights Register *x* registers depicted in Figure 18, Figure 19 and Figure 20. Currently, each weight is an 8-bit field. Each input of the MCCU maps directly to the outputs of the crossbar. Thus, the weight for the MCCU input 0 corresponds to the signal in crossbar output 0.

*RDC*

The Request Duration Counter or RDC depicted in Figure 21 is comprised of a set of 8-bit counters and comparators that allow monitoring the length of a CCS signal, recording the number of clock cycles of the longest pulse and comparing this number with the defined weight.

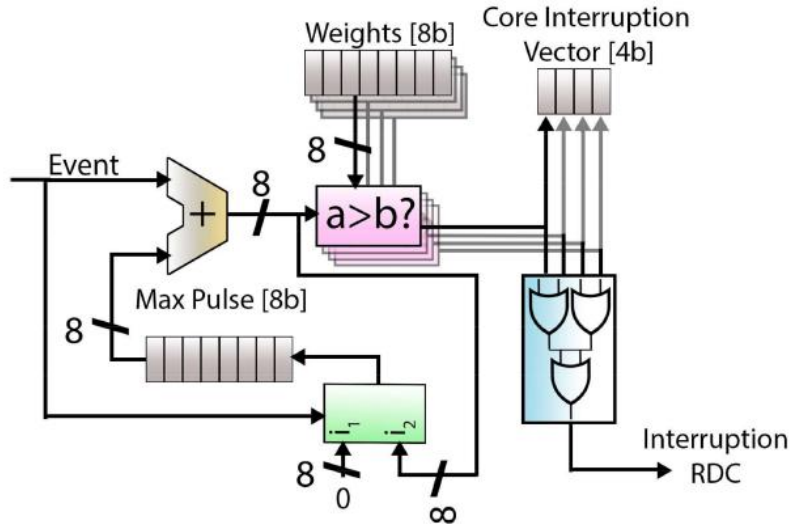


Figure 21 SafeSU - Block diagram of the RDC mechanism

The current release provides monitoring for crossbar outputs 0 to 7. The weights for each signal are shared with the MCCU and are stored in the RDC Event Weights Register  $x$  registers depicted in Figure 23. Weights are 8-bit fields. Counters have overflow protection, preventing the count from wrapping over the maximum value. The maximum value for each event (watermarks), is stored in the RDC Watermark Register  $x$  registers depicted in Figure 24.

The RDC shares the main configuration register with the MCCU (Figure 18). Through this register, the unit can be reset and enabled through the corresponding fields. Such fields are active high signals.

The unit does provide access to the internal interrupt vector (Figure 22), but such information is redundant and may be removed in future releases. Given the current watermarks and assigned weights, the events responsible for the interrupt can be identified. The RDC interrupt has been routed to AHB interrupt 11.

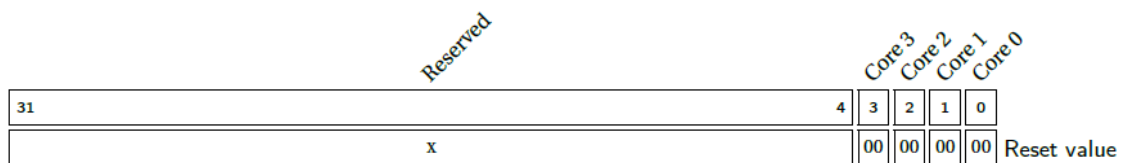


Figure 22 SafeSU - RDC Interrupt Vector (0x0A0)

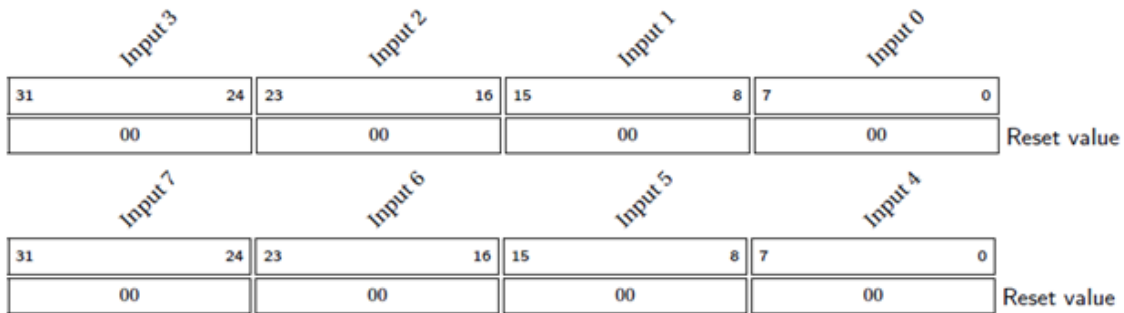


Figure 23 SafeSU - RDC Event Weights Registers 0 and 1 (shared with MCCU; 0x098, 0x09C)

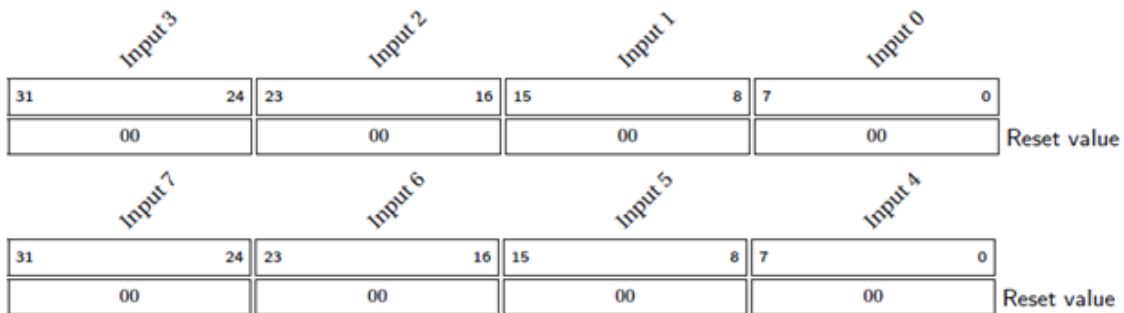


Figure 24 SafeSU - RDC Watermark Registers 0 and 1 (0x0A4, 0x0A8)

## 6 SEMANTIC MANAGEMENT OF THINGS AND EMBEDDED AI AND NLP

### 6.1 SUPPORT TO INTEROPERABILITY ALONG THE STACK

Semantic management encompasses a wide range of concepts aimed at ensuring that digital assets can be consistently described, understood, and processed across systems and organizations. Within this broad scope, this section focuses on the crucial concept of *interoperability*.

Supporting interoperability involves a spectrum of singular efforts, from facilitating the exchange of clear and comprehensive documentation to providing technical components that mediate between heterogeneous services or transform data into the required formats and specifications. Different layers of the IT stack have different requirements when dealing with the concept of interoperability. We decided to perform our analyses on the following stack definition:

- Hardware
- Networking
- Deployable software
- Services
- Datasets
- AI models

For each of such layers we started with two simple questions, “what does it mean interoperability in this context?” and “how we can ease interoperability?”. The initial answers range from “ease the exchange of technical documentation” to “perform data integration and service mediation”. All those answers denote a common enabler for interoperability, namely the ability to describe assets, ranging from hardware and software components to datasets and digital services, in a structured and unambiguous manner. This structured description is essential for enabling their effective integration within and across technical environments.

Once a common metadata model is defined, a catalogue of assets can be put in place (and it will be an outcome of WP5) to gather assets descriptions and to link them. The catalogue can then become the command and control center for operational interoperability, which will be also addressed in the context of WP3. By operational interoperability we mean the possibility to put in place data converters and service mediators to ease the integration of software and services implemented independently by different organizations.

In the following, we will explore mechanisms and practices that support such interoperability and that is going to implement in the context of SMARTY, and how they contribute to more seamless and scalable integration processes.

#### 6.1.1 Interoperability-oriented metadata ontology

##### *Background on semantic web technologies*

The semantic web (SW), an extension of the web, allows data to be machine-readable - that is, interpretable and processable by computers - by encoding its semantics [Grigoris2004]. As a result, computers can understand not just the structure of data but also the meaning of its content. The World Wide Web Consortium (W3C) is responsible for the standardization of semantic web technologies. One of its key specifications, the Resource Description Framework (RDF), provides a model for structuring information. RDF identifies web resources using unique web identifiers (URIs) and describes them through statements - known as triples - comprising a subject, predicate, and object. These triples form a graph structure, where nodes represent resources or values, and edges define semantic relationships between them. RDF Schema (RDFS) extends the basic RDF vocabulary, while the Web Ontology Language (OWL) introduces a more expressive, logic-based framework for defining complex knowledge structures and relationships. Together, these three standards enable the formal definition of domain-specific concepts and the relationships between them [Gruber1993]. A formally defined, shared vocabulary that describes a domain is known as ontology. When an ontology is applied to a set of specific instances - represented as nodes and edges in a graph - it results in a Knowledge Graph (KG), which instantiates the concepts and relationships defined within the ontology.

##### *Methodology*

This section outlines the methodology we adopt for constructing the metadata ontology, that is the Linked Open Terms (LOT) methodology<sup>1</sup> [Poveda2022], a lightweight approach designed for creating ontologies within the industry domain. As illustrated in Figure 28, the LOT

---

<sup>1</sup> <https://lot.linkeddata.es/>

methodology consists of iterative cycles based on a workflow that includes the following key activities: (i) ontology requirements specification, (ii) ontology implementation, (iii) ontology publication, (iv) ontology maintenance.

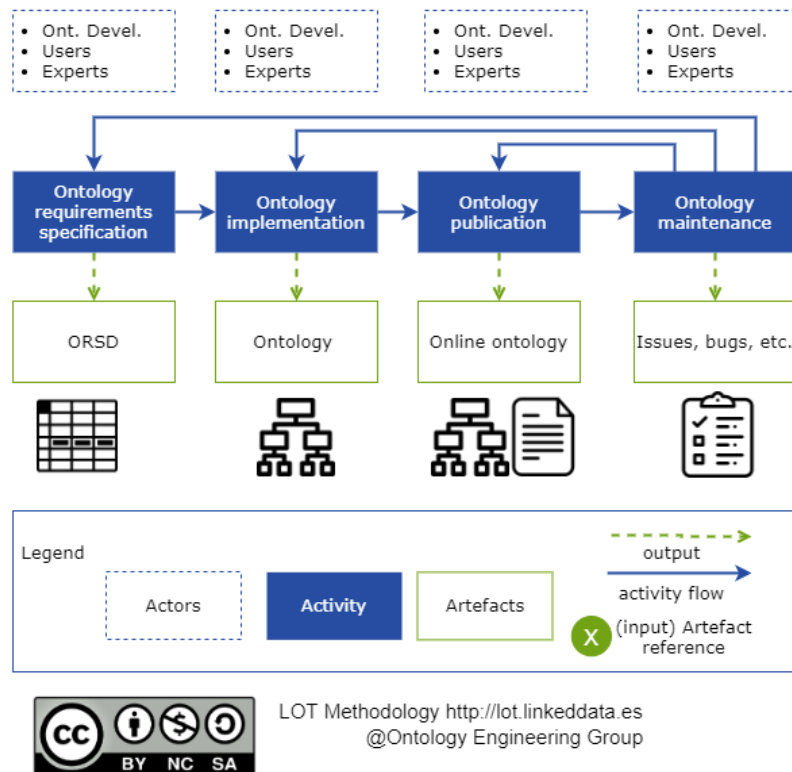


Figure 25. LOT methodology for ontology development

**Ontology Requirements Specification.** The ontology requirements specification phase involves defining the purpose and scope of the ontology, as well as outlining the specific requirements it must fulfill. To achieve this, the ontology designers must define a set of scenarios requiring data to be modelled within the ontology. Additionally, they may gather relevant documents and domain-specific resources from domain experts and data owners. From such materials, functional requirements can be derived in the form of competency questions (CQs), which are questions the ontology should be able to answer [Gruninger1995], along with descriptive facts - natural language statements that define domain-specific entities and terminology. Collaboration between ontology engineers, domain experts, and end users is crucial in this phase to ensure the accuracy and completeness of the gathered requirements.

- **Ontology Implementation:** Before constructing the ontology with a formal language, it should be developed as preliminary conceptual model that informally represents the knowledge domain. This model usually includes a proposed set of concepts, relationships, and their hierarchies. Then, such model is encoded and expanded using the OWL ontology language. As a best practice in ontology development, it is recommended to reuse existing ontological resources in order to support interoperability [Carriero2020].
- **Ontology publication:** The goal of this activity is to ensure the ontology is accessible and well-documented online. After selecting an appropriate license, incremental releases can be published online, possibly in multiple syntaxes (e.g., Turtle, RDF/XML, JSON-LD,



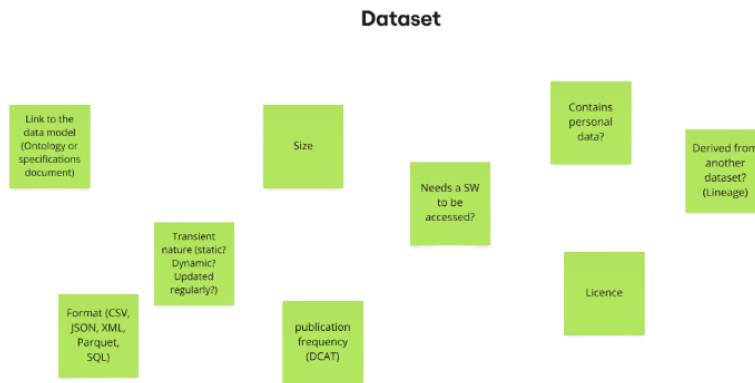


Figure 27. Example of requirements collected w.r.t. relevant metadata of assets (e.g., dataset)

Building on the results of the workshop, the team of ontology designers worked to extract an initial set of requirements from them, in the form of competency questions accompanied by facts focusing on the domain concepts that emerged. Each proposed competency question is associated with an identifier, provenance information (e.g., from which post-it it comes from), and examples of data that would help answer the question. Our initial set of requirements includes 19 competency questions on hardware, 27 competency questions on software, while the extraction of CQs on the other types of assets is still ongoing; and 32 facts about hardware and software. Such requirements will undergo a process of validation by domain experts, so that they can be used for developing the ontology. Future iterations may lead to the definition of additional CQs and facts.

Let us make an example of how an input from the workshop translates into competency questions and facts relevant for developing the ontology.

From the input “firmware version” in the hardware frame, we derived the following 3 CQs:

- “Which is the firmware of the hardware?”
- “Which is the version of the firmware required by the hardware?”
- “Which is the version of the software?” (which generalizes over the initial input)

Along with 3 facts that detail what we mean by the term firmware, hardware, and firmware/software version in our ontology. Such CQs will drive the development by allowing us to support the representation of the relation between a firmware associated with a hardware, and its versioning metadata.

#### *Initial outcomes of analysis of state-of-the-art relevant schemas and ontologies*

As a preliminary activity to the development of the ontology, and based on our requirements, we performed an initial analysis of existing ontologies that could be reused, specifically addressing metadata on all kinds of assets, and hardware and software characteristics.

**Metadata schemas and ontologies.** **DCAT** (Data Catalog Vocabulary)<sup>2</sup> is an RDF vocabulary developed to enhance interoperability between data catalogues published on the web. Indeed, it supports the representation of metadata about different kinds of resources, like catalogues,

<sup>2</sup> <https://www.w3.org/TR/vocab-dcat-3/>

catalogue resources, datasets, and data services. DCAT, along with creating novel classes and properties, e.g., keyword and landing page, also reuses other existing ontologies, including **DCMI Metadata Terms**<sup>3</sup>, **ADMS**<sup>4</sup> (Asset Description Metadata Schema), and **PROV-O**<sup>5</sup> (Provenance Ontology). DCMI Metadata Terms is a general-purpose metadata vocabulary for describing resources of any type, and includes very general properties like title and description, and more specific ones, like access rights and publisher. ADMS is a profile of DCAT for describing (semantic) assets with metadata like file, license, repository, status, etc. PROV-O models provenance information generated in different systems and under different contexts; examples of relevant ontology entities reused in DCAT are activity and attribution, for representing, for instance, when an entity was generated by an agent in the context of a certain activity. The Data Product Ontology<sup>6</sup> (**DPROD**) is another profile that extends DCAT and has been designed to describe and manage data products (i.e., data as assets to be managed and distributed like any other product) and data services consistently across platforms, modelling metadata properties like the data product owner, the input and output port of the set of services exposed by a data product, etc. A relevant open-source metadata model, formalized as a YAML schema, is the **Open Data Product Specification**<sup>7</sup>, which defines objects and attributes related to digital data products. It is based on existing standards (Schema.org) and best practices. **Schema.org**<sup>8</sup>, aimed at enabling structured data markup on web pages, includes a number of types and properties based on the original DCAT work and supports the representation of datasets and data catalogues.

*Hardware-related ontologies.* The Semantic Sensor Network (**SSN**) ontology<sup>9</sup> describes sensors and their observations, the related procedures, the features of interest that are studied and observed, the samples used to do so, and the observed properties. SSN includes a lightweight self-contained core ontology named **SOSA**<sup>10</sup> (Sensor, Observation, Sample, and Actuator) for its basic classes and properties. SSN reuses the foundational Dolce+DnS Ultralite (DUL) ontology<sup>11</sup> and the related **Stimulus-Sensor-Observation Ontology Design Pattern** [Janowicz2010]. Even if it is an ontology designed for sensors, SSN also includes more general hardware-related entities, like system, system property, system capability, etc. The IT Service Management Ontology<sup>12</sup> (**ITSMO**) describes resources related to IT Service Management best practices, and enables the interchange of specific pieces of information between systems. It extends **PROV-O** and **Schema.org** (see above), and addresses both the hardware and the software domains, e.g., a configuration item and a system component can refer to either hardware or software. Relevant properties link software/hardware/documentation, etc., to agents (e.g., has owner, has responsible) and to other objects (e.g., dependencies between configuration items, the endpoint of a service, criticalities of an entity, etc.).

*Software-related ontologies.* As described in the previous paragraph, the **ITSMO** ontology partly addresses software-related attributes. The **Software Description Ontology**<sup>13</sup> describes software components, including their metadata (attribution, licensing, usage instructions, etc.) and their

---

<sup>3</sup> <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/>

<sup>4</sup> <https://www.w3.org/TR/vocab-adms/>

<sup>5</sup> <https://www.w3.org/TR/prov-o/>

<sup>6</sup> <https://ekgf.github.io/dprod/>

<sup>7</sup> <https://opendataproductions.org/>

<sup>8</sup> <https://schema.org/>

<sup>9</sup> <http://www.w3.org/ns/ssn/>

<sup>10</sup> <http://www.w3.org/ns/sosa/>

<sup>11</sup> <https://www.loa.istc.cnr.it/ontologies/DUL.owl>

<sup>12</sup> <https://w3id.org/itsmo>

<sup>13</sup> <https://w3id.org/okn/o/sd#>

inputs, outputs and variables. The ontology extends **Schema.org** (see above) and Codemeta<sup>14</sup> vocabularies, and builds on a previously developed ontology, namely **OntoSoft**<sup>15</sup>, which is used by the OntoSoft portal<sup>16</sup> to organize its metadata. **Codemeta** includes all basic software attribution terms, such as author, maintainer, funding, license, related publications, etc. The Software Description Ontology is also documented with its requirements, expressed in the form of competency questions<sup>17</sup>. The Software Ontology<sup>18</sup> (**SWO**) describes software tools, their types, tasks, versions, licensing, and provenance, and has been designed in the context of the OBO Foundry (Open Biological and Biomedical Ontology Foundry), a community that develops interoperable ontologies for the biological sciences. SWO contains detailed information on licensing and formats as well as software applications, mainly related to the bioinformatics community. The **Core Ontology of Software**<sup>19</sup> formalizes the most fundamental concepts which are required to model both software components and web services, such as software, data, users, access rights, interfaces, etc. It reuses the foundational DOLCE ontology<sup>20</sup> along with the ontology design patterns (ODPs) extracted from it. The **SPDX ontology**<sup>21</sup> is the ontological version of the SPDX language<sup>22</sup>, that is an open standard for communicating software bill of materials (SBOM) information, such as software components, licenses, copyrights, and security references. Another language for representing SBOMs is CycloneDX, which is also consumed and produced by the Dependency-Track<sup>23</sup> intelligent component analysis platform. As for hardware and software behind an AI model, model cards reports (such as Google's Model Cards) provide a description of machine learning models with information about their evaluation, limitations, intended use, etc., and offer a wide range of information that could be collected. The Model Card Report Ontology<sup>24</sup> (**MCRO**) aims at capturing the conceptual-level information embedded in model card reports, and reuses OBO Foundry ontologies, like the previously mentioned SWO. Along with properties common to other ontologies we analyzed (such as owner, version, license), MCRO models also performance metrics (like accuracy), evaluation/training data, ethical considerations and risks, etc. AIRO<sup>25</sup> (AI Risk Ontology) is an ontology for expressing risk of AI systems based on the requirements of the AI Act<sup>26</sup>. It includes relevant concepts such as AI system (linked with its domain, capabilities, purpose, users, providers, license, version, etc.), hardware platform, risk, vulnerability, and severity, testing/training/validation data, etc. VAIR<sup>27</sup> (Vocabulary of AI Risks) is intended to assist with the identification and documentation of risks by providing a common open vocabulary for AI risks

---

<sup>14</sup> <https://codemeta.github.io/terms/>

<sup>15</sup> <https://ontosoft-earthcube.github.io/ontosoft/ontosoft%20ontology/v1.0.1/doc/index.html>

<sup>16</sup> <https://www.ontosoft.org/portal/#list>

<sup>17</sup> [https://figshare.com/articles/dataset/Requirements\\_for\\_the\\_Software\\_Description\\_Ontology\\_May\\_2019/\\_9249470?file=16838615](https://figshare.com/articles/dataset/Requirements_for_the_Software_Description_Ontology_May_2019/_9249470?file=16838615)

<sup>18</sup> <https://github.com/allysonlister/swo>

<sup>19</sup> <https://km.aifb.kit.edu/sites/cos/>

<sup>20</sup> <https://www.loa.istc.cnr.it/dolce/overview.html>

<sup>21</sup> <http://spdx.org/rdf/terms>

<sup>22</sup> <https://spdx.dev/use/specifications/#current-version>

<sup>23</sup> <https://dependencytrack.org/>

<sup>24</sup> <http://github.com/UTHealth-Ontology/MCRO>

<sup>25</sup> <https://w3id.org/airo#>

<sup>26</sup> <http://data.europa.eu/eli/reg/2024/1689/oj>

<sup>27</sup> <https://delaramglp.github.io/vair/>

## 6.1.2 Service mediation through declarative semantic-web based mappings

*The problem of service mediation*

Data and service integrations are a crucial problem to address as different partners and respective software systems will need to collaborate to address the needs of the project's use cases. The problem of *data integration* arises because different systems deal with various data formats, meaning that there is a syntactic mismatch between systems. Additionally, there is often an additional semantic mismatch between the data generated and used in different systems. This is natural, as the data semantic develops according to the special needs of the system and cannot consider the semantics employed by a different system developed in isolation. As an example, different systems may model the concept of a "length", but one system may use XML while the other uses JSON to transmit this information. This is an example of syntactic mismatch. Additionally, if not explicitly declared in the data itself, the semantic of the length concept might be different, as in system number one it is expressed in meters while in system number two it is expressed in feet.

The problem of *service integration* is similarly important, as two systems may use different communication protocols and as such might need a third system to mediate communication between them. For example, one system may only be able to communicate via MQTT<sup>28</sup> while a second system might only be able to receive messages via the NATS<sup>29</sup> protocol. To connect these systems to a third one can be deployed to receive MQTT messages and then forward these messages via NATS to the appropriate receiver. Of course, this means that this third system must be able to support a wide gamut of communication protocols to prove effective. This is an example of service integration.

We make use of the term *service mediation* to refer to an approach that solves both previously described problems. This is because data and service integration problems often arise together, meaning that to connect different systems it is necessary to mediate at the communication protocol level and at the syntactical and semantical level.

To address the syntactical and semantical facets of data integration problems we will follow an *any to one* mapping (data conversion) approach, which is best defined by comparing to a more traditional *any to any* mapping strategy both of which are shown in Figure 28. In an any-to-any mapping approach, each system must translate its data to and from every other system, leading

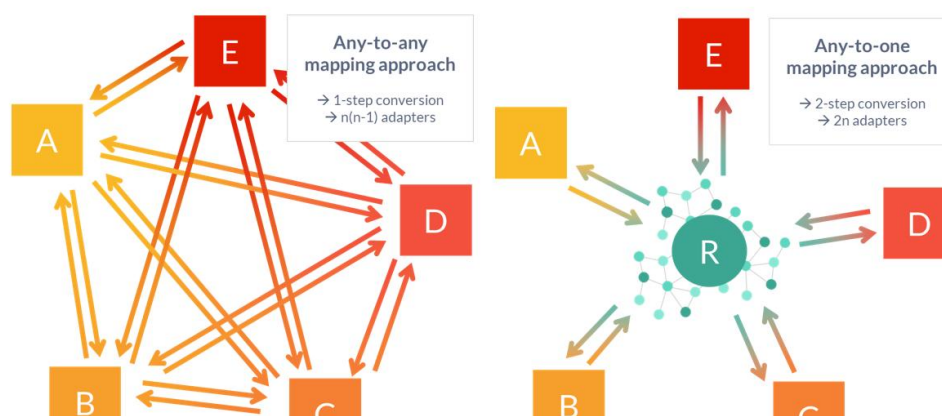


Figure 28: Any-to-any versus any-to-one mapping approaches

<sup>28</sup> <https://mqtt.org/>

<sup>29</sup> <https://docs.nats.io/>

to a high number of mappings. In contrast, the any-to-one mapping approach reduces the number of required mappings by introducing a use case or task specific reference model, with each system mapping its data to and from this central model. This reference model should be capable of defining all the concepts that are needed by the integrated systems.

We will use semantic web technologies to formulate these reference models. This is because expressing these reference models through the resource description framework (RDF), described in Section 6.1.1, offers multiple benefits. First, it is a machine-readable standard for encoding semantics through appropriately selected vocabulary. This means that a reference model declared through RDF can be viewed as reference ontology, meaning that the semantic is fixed and can be considered as a fixed point of truth. Second, data expressed in RDF forms a knowledge graph whose schema can be easily extended should the need arise. Finally, the RDF knowledge graph can be queried via a standardized SPARQL<sup>30</sup> protocol to extract relevant subsets of information.

With this RDF based any-to-one approach we can further break down the problem in two distinct *lifting* and *lowering* steps showcased in Figure 29. The lifting step is necessary to convert from a data format, e.g. JSON to RDF. At this step the semantic is fixed to the one specified in the reference ontology. The lowering step is the opposite operation, where from RDF we convert to the data format needed by the appropriate data recipient. Crucially, the data obtained by the lowering step follows the semantic of the reference ontology.

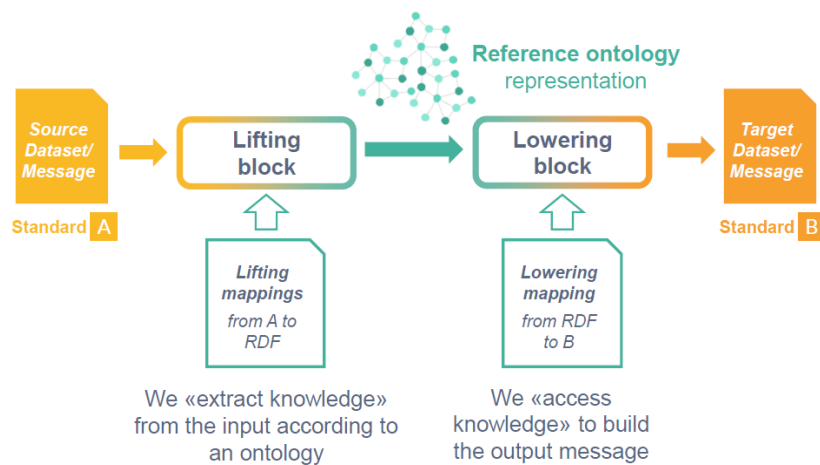


Figure 29: Lifting and lowering steps for the RDF Any-to-One Mapping Approach

#### Components used to implement service mediation

To implement service mediation, Cefriel will use the open source internally developed *Chimera*<sup>31</sup> software. Chimera is a Java-based, open-source framework built on *Apache Camel*<sup>32</sup> for performing service mediation through semantic data transformation and fusion pipelines.

<sup>30</sup> <https://www.w3.org/TR/sparql11-query/>

<sup>31</sup> <https://github.com/cefriel/chimera>

<sup>32</sup> <https://camel.apache.org/>

By leveraging Apache Camel, Chimera supports the service interoperability required for connecting systems through a wide array of communication protocols, such as HTTP, WebSocket, and MQTT implemented as Apache Camel components. Chimera extends Apache Camel by defining additional components for handling RDF data to enable the semantic-web based any-to-one approach described in the previous section.

The Chimera defined components are:

- **Graph Component:** Uses the RDF4J<sup>33</sup> library to perform RDF graph operations including:
  - *Graph Get/Add:* Accessing and augmenting RDF graphs.
  - *Graph Construct:* Executing SPARQL CONSTRUCT queries.
  - *Graph Select:* Executing SPARQL SELECT queries.
  - *Graph Ask:* Executing SPARQL ASK queries.
  - *Graph Inference:* Supporting RDFS-based inference.
  - *Graph SHACL:* Validating RDF graphs using SHACL shapes.
  - *Graph Detach/Dump:* Cleaning or exporting graphs to formats like Turtle or N3.
- **RML Component:** Enables the execution of RML mappings for lifting data from heterogeneous sources into RDF, powered by an optimized version of RMLMapper<sup>34</sup>.
- **Mapping Template Component:** Enables the execution of mappings for the lifting and lowering operations written using the Mapping Template Language<sup>35</sup> (MTL), providing performance improvements and more fine-grained control over mappings written using RML.

## 6.2 EMBEDDED AI AND NLP

There are topics strictly related to the interplay between hardware and software, such as AI and Natural Language Processing. This task focuses on designing and evaluating a Knowledge Graph-based cooperative perception system that integrates multimodal sensor data, vehicular signals, and environmental context using semantic web technologies. A central objective is ontology alignment and integration, where standards like SOSA, VSS, and OpenXOntology are mapped into a unified schema to represent real-time driving scenes. This involves resolving semantic ambiguities, developing an ontology fusion layer, and enabling machine-interpretable perception graphs. The system combines symbolic reasoning (e.g., OWL rules, SHACL constraints, SPARQL queries) with LLM-based perception synthesis, facilitating tasks such as summarizing graph states, generating temporal event descriptions (e.g., "vehicle approaching fast from left blind spot"), and supporting cooperative intent prediction. These language-based insights are then translated back into graph structures for downstream reasoning and decision-making, creating a seamless bridge between perception and planning. Simulation and validation will be conducted using environments like CARLA, enhanced with multimodal datasets that

---

<sup>33</sup> <https://rdf4j.org/>

<sup>34</sup> <https://github.com/RMLio/rmlmapper-java>

<sup>35</sup> [https://github.com/cefruel/mapping-template/wiki/Mapping-Template-Language-\(MTL\)](https://github.com/cefruel/mapping-template/wiki/Mapping-Template-Language-(MTL))

mimic real-world cooperative contexts. Embedded AI models will process sensor streams on virtual edge devices, while LLMs will be evaluated for tasks such as onboard situation reporting, maneuver explanation, and interactive planning dialogues. Performance metrics include system responsiveness, graph update quality, LLM grounding accuracy, and cooperative inference reliability.

To further strengthen the project, Knowledge Graph Foundation Models (KGFM) will be used for data alignment. During pre-training, KGFM tools like ULTRA will generate low-dimensional embeddings for entities and relations, capturing both semantic context and structural neighborhood information. This unified representation will aid in aligning and organizing data across the knowledge graph. Efforts to inject semantics into KGFM using LLMs will complement this, enhancing the overall semantic richness and usability of the system.

Additionally, optimized algorithms for local search on graphs will be developed to improve Retrieval Augmented Generation (RAG) systems. This will involve combining token-level embeddings for entities and relations with query and content-level embeddings, supported by LLM-based searching and re-ranking techniques. These algorithms will be evaluated using the knowledge graphs developed within the project, aiming to enhance the efficiency and accuracy of graph-based data retrieval and reasoning.

In summary, this collaborative effort aims to create a robust, semantically rich, and efficient knowledge graph-based cooperative perception system, integrating advanced AI-driven reasoning, alignment techniques, and validation methodologies.

## 6.3 HARDWARE/SOFTWARE INTEGRATION

As explained before, one of the tasks in WP3 is dealing with interoperability, which also means integrating software artifacts and hardware artifacts. In the following sections we'll document the initial aspects that will be included in the ontologies related to the low levels of the stack (hardware and networking) and their integration with low level software artifacts. Whenever the "Asset #XXX" notation is used, is referring to D2.1 document naming convention.

### 6.3.1 Semantic Management of Quantum-Resistant Secure Element (Asset #2)

For the Quantum-Resistant Secure Element (QR-SE), the ontology is designed to encompass both hardware and software aspects to ensure comprehensive understanding and integration of its components and subsystems. Below is the detailed explanation of the ontology framework for the QR-SE:

Hardware Dimensions:

- The hardware aspects of the QR-SE are critical as they define the physical and computational capabilities that enable secure cryptographic operations. The following categories are included:
- Supported Classical Algorithms: The QR-SE must support conventional cryptographic algorithms such as RSA and ECC. These algorithms remain relevant for backward compatibility with legacy systems and hybrid cryptographic approaches.

- **Supported QR-Algorithms:** As the QR-SE is designed with post-quantum security in mind, it supports quantum-resistant algorithms like KYBER (for key exchange) and DILITHIUM (for digital signatures). The inclusion of these algorithms ensures the system is resistant to potential quantum computing attacks.
- **Serial Interfaces:** The hardware interfaces supported by the QR-SE, such as I2C and SPI, define how the secure element communicates with external components and subsystems. These interfaces ensure compatibility with various host environments.
- **Performance Metrics:** Performance is often characterized using metrics such as the number of signatures or cryptographic operations the QR-SE can process per second. This metric is critical for evaluating the suitability of the secure element in performance-sensitive applications.
- **Chip Package:** The chip package refers to the physical form factor of the QR-SE and its encapsulation. It determines the deployment constraints, such as size, thermal performance, and robustness.
- **Power Consumption:** The power consumption of the QR-SE is a key consideration for energy-constrained applications, such as IoT devices. It reflects the efficiency of the hardware design and its impact on the overall system.

#### Software Dimensions:

The software aspects of the QR-SE are equally important as they define the functional behaviour, compatibility, and ease of integration with higher-level applications. These aspects include:

- **JavaCard Applet Version:** For secure elements that utilize JavaCard technology, the applet version indicates the compatibility with the JavaCard platform and standards. This ensures interoperability with existing infrastructures.
- **Software License:** The QR-SE software components and applets must include clear licensing terms to comply with intellectual property regulations and ensure proper usage in commercial or open-source projects.
- **Toolchain:** The toolchain includes all software tools and environments used to develop the QR-SE, such as the compiler, JavaCard OS version, and associated SDKs. These tools ensure the reliability and maintainability of the software.
- **Implemented API and Communication Interfaces:** The software must expose Application Programming Interfaces (APIs) and communication protocols (e.g., APDU commands for smart cards) to allow seamless integration with host systems and applications.
- **Applet-Implemented Cryptographic Functionality:** This aspect describes the cryptographic operations implemented within the JavaCard applet, including secure key management, encryption, decryption, and signature generation. The functionality should align with the supported classical and QR algorithms.

#### 6.3.2 Semantic Management of Ultra-low Power Processor for PQC and Edge-AI (Assets #1 & #24)

The ontology for the Ultra-low Power Processor for PQC and Edge-AI, addresses both hardware and software components. This ensures a comprehensive and integrated understanding of the system architecture.

#### Hardware [Asset #1]:

This section details the hardware features encompassed within the ontology framework. The hardware aspects define the physical and computational capabilities that enable secure cryptographic operations in edge environments. The following categories are included:

- **Documentation:** Complete hardware documentation is provided to facilitate integration with other systems. Datasheets include critical information such as available interfaces, connectors, ports, power supply requirements, setup instructions, and configuration options.
- **Hardware and Firmware Version:** Information about the hardware and firmware versions ensures compatibility across different software components and driver versions. Tracking these versions is essential for maintaining system integrity and performance.
- **Performance Metrics:** The performance of the Hardware is characterized by data rates and the number of signatures per second. These metrics vary depending on the communication interface used (e.g., UART, CAN, I2C, SPI) and are constrained by the physical properties of each interface. The knowledge of these limits is crucial for ensuring compatibility with the target system.
- **Power consumption:** Power consumption is a critical factor for battery-powered IoT devices or systems requiring low energy usage. Depending on the specific use case, power consumption can be optimized by enabling or disabling different features. The SoC includes three independent cores, each of which can be selectively activated based on performance and power requirements.
- **Supported Cryptographic Algorithms:** The processor supports cryptographic operations based on the QR-SE algorithm developed by IFAG. The cryptographic capabilities depend on the available applets and implemented algorithms, allowing customization based on security needs.
- **Available Interfaces:** The hardware includes physical communication interfaces such as UART, SPI, and I2C by default. Additional interfaces, such as CAN or Ethernet, can be incorporated if required by the use case. For wireless communication, BLE support is available. Internally, the processor implements both I2C and IEC-7816 interfaces as part of its QR-SE integration.
- **Communication protocol and drivers:** The communication protocols, including connection establishment procedures for different interfaces, are thoroughly documented. Drivers for specific platforms can be provided, and partners can use the provided documentation to implement custom drivers as needed.

#### Software [Asset #24]:

The software components of the Ultra-low Power Processor for PQC and Edge-AI devices define its functional behavior, system compatibility, and ease of integration with higher-level applications. The ontology captures the following key aspects:

- **Communication protocols and interfaces:** The firmware for the PSoC Edge platform implements the Hardware Abstraction Layer (HAL) for the I2C and IEC-7816 interfaces, facilitating communication with the integrated QR-SE module. On top of this communication layer, the Secure Element (SE) drivers provided by IFAG are integrated. For external interfaces such as UART, CAN, and SPI, the firmware implements the HAL

and communication protocols using a modular, layer-oriented architecture to promote scalability and maintainability.

- **Application Layer:** The firmware is designed to be adaptable based on specific use cases (e.g., Use Case 1 and Use Case 5). In both cases, the primary objective of the hardware-software system is to establish and maintain trustworthy communication between Edge nodes, ensuring data integrity and security.
- **Toolchain:** For development, maintenance, and future enhancements, the full toolchain information, including the tested GCC compiler version and associated tools, will be provided. This ensures that partners and developers can replicate the development environment for consistent builds and system behavior.
- **Cryptography functionality:** The firmware leverages the cryptographic capabilities of the QR-SE module, particularly utilizing the KYBER and DILITHIUM algorithms. These functionalities include signature generation and encryption, allowing external systems to benefit from post-quantum cryptographic operations according to the specific use case requirements.
- **Drivers and API:** Drivers tailored to specific platforms will be developed based on use case requirements. Comprehensive API documentation will be provided to facilitate integration with external systems, enabling efficient communication and control of the cryptographic hardware features.
- **Firmware Versioning:** Firmware version information will be maintained and documented to ensure compatibility between hardware, drivers, and software layers. Proper version tracking helps manage updates and ensures stable system integration.
- **Licensing:** All firmware and driver components will include clear licensing terms, ensuring compliance with intellectual property regulations. These terms define the conditions for commercial and open-source usage, safeguarding both developer and user rights.

### 6.3.3 Semantic Management of Programmable Data Plane Testbed (Asset #10)

Use Case 2 will be validated leveraging the heterogeneous facilities included in the CNIT testbed, located in Pisa (Italy). The testbed, described in D2.1 as asset #10 includes computing and network equipment, enabling the emulation of metro-edge infrastructures including transport optical nodes, wired and wireless access networks, edge and cloud computing resources.

The following HW will be specifically exploited for Use Case 2 validation:

- **Network resources:**
  - Network Switch: P4 Programmable 100G switches
  - Data Processing Unit (DPU): NVIDIA BlueField operating at up to 200Gb/s
- **Computing resources:**
  - n.8 DELL PowerEdge with GPU
  - Prototype of edge micro datacenter (EMDC) developed in the Chips JU (KDT) BRAINE project. The prototype encompasses an advanced monitoring system able to retrieve accurate HW conditions, including power consumption.

In order to execute SMARTY solutions over the testbed resources, the definition of a semantic description of the aforementioned key HW elements of the asset is required to support interoperability while guaranteeing the effective usage of heterogeneous resources.

#### 6.3.4 Semantic Management of Post Quantum Computing Accelerator (Asset #4)

The PQC Accelerator developed by BSC is integrated into the SELENE SoC to offload post-quantum cryptographic operations (ML-KEM and ML-DSA). It is implemented using High-Level Synthesis (HLS) tools and supports modular pipelined execution through AXI-Lite and AXI-Full interfaces. To support transparent system integration and verification, a semantic layer has been introduced.

This layer describes key elements of the accelerator, such as register configurations, operation modes, I/O memory regions, and execution states, using an ontology-based approach. It allows automated reasoning about which cryptographic primitive is active, whether an operation has completed, and how to route data securely across the SoC.

The semantic description also helps track performance and compliance with security policies. For example, it can indicate if a PQC operation is delayed due to memory contention, or if input/output buffers are correctly aligned before execution.

This integration validates **Use Case 1** by enabling secure key exchange and digital signature operations for on-board electronics. The semantic layer improves system observability, supports automated checks during OTA updates, and can work in conjunction with other monitoring units like **SafeSU** to ensure timing and resource guarantees.

#### 6.3.5 Semantic Management of Safe Statistics Unit (SafeSU) (Asset #6)

Safe Statistics Unit (SafeSU) will be validated together with Use Case 1 using the SELENE platform. The SafeSU will be integrated into the SELENE platform and software drivers will also be developed and will be integrated into the Use Case software.

The SafeSU will monitor and leverage the throughput of the AXI-4 bus of the different masters including the PQC accelerator (Asset #7) as well as the general-purpose cores of the platform. Using a quota-based mechanism to limit the timing-impact that AXI-4 accelerators can apply to the critical-application (Use Case 1) running on the general-purpose cores.

## 7 CONCLUSIONS

---

The D3.1 First Report on Hardware Accelerators for Secure Communications lays crucial groundwork for the SMARTY project by addressing the pressing demand for secure, high-performance communication systems. It highlights the role of hardware accelerators in tackling modern security challenges and positions quantum-resistant secure elements and transceiver as a forward-looking solution.

The report's focus on ontology-driven methodologies ensures that the developed technologies are structured, interoperable, and aligned with real-world requirements. This collaborative and iterative approach not only strengthens the project's outcomes but also aligns them with evolving industry standards.

As the SMARTY project progresses, the findings and methodologies outlined in this report will serve as a key reference for future phases. By combining technical innovation with practical application, the project is well-positioned to set new benchmarks in secure communication systems, ensuring resilience and scalability in an interconnected and increasingly challenging digital environment.

## 8 REFERENCES

---

- [Carriero2020] V.A. Carriero, M. Daquino, A. Gangemi, A.G. Nuzzolese, S. Peroni, V. Presutti, F. Tomasi, “The landscape of ontology reuse approaches”, Applications and practices in ontology design, extraction, and reasoning, IOS Press, 2020, pp. 21-38
- [Grigoris2004] A. Grigoris, F. Van Harmelen. “A Semantic Web primer”. MIT press, 2004.
- [Gruber1993] T.R. Gruber, “A translation approach to portable ontology specifications” Knowledge acquisition, 5.2, 1993, pp. 199-220.
- [Gruninger1995] M. Gruninger, M.S. Fox. “The role of competency questions in enterprise engineering.” Benchmarking - Theory and practice, MA: Springer US, 1995, pp. 22-31
- [Janowicz2010] K. Janowicz, M. Compton, “The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology”, *SSN*, 668.
- [Poveda2022] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López, R. García-Castro, “LOT: An industrial oriented ontology engineering framework”, Engineering Applications of Artificial Intelligence, 111, 104755, 2022