Cross-platform **O**pen **S**ecurity **S**tack for **Con**nected Device

D5.2 Integrated CROSSCON Security Stack - First Version

Document Identification			
Status	Final	**Due Date**	31/07/2024
Version	1.0	**Submission Date**	31/07/2024

Related WP	WP5	**Document Reference**	D5.2
Related Deliverable(s)	D1.4, D1.6, D2.1, D3.1, D3.2, D4.1, D4.2	**Dissemination Level (*)**	PU
Lead Participant	CYSEC	**Lead Author**	Yannick Roelvink
Contributors	3MDEB, BIOT, TUD, UMINHO, UWU	**Reviewers**	Ziga Putrle (BEYOND)
			Gergely Eberhardt (SLAB)

Keywords:
CROSSCON Stack Demonstrator, Integration, Use Case Prototypes

Document Information

List of Contributors	
Name	Partner
Yannick Roelvink	CYSEC
Emna Amri	CYSEC
Artur Raglis	3MDEB
Tymoteusz Burak	3MDEB
Shaza Zeitouni	TUD
Ainara García Barinaga	BIOT
David Puron	BIOT
David Cerdeira	UMINHO
Lukas Petzi	UWU

Document History			
Version	Date	Change editors	Changes
0.1	03/07/2024	Yannick Roelvink	Initial version, document layout definition and allocation of sections to contributors. Included first draft of Introduction chapter.
0.2	15/07/2024	Artur Raglis Yannick Roelvink	Integration of first version of UC1 prototype sections, first version of Chapter 2 - CROSSCON Stack Development Progress.
0.3	16/07/2024	Shaza Zeitouni	Contribution to Chapter 2 & inclusion of UC5 prototype section.
0.4	17/07/2024	Ainara García Yannick Roelvink	Inclusion of UC2, UC3 & UC4 prototype sections.
0.5	18/07/2024	Yannick Roelvink	Reformatting of UC descriptions, initial version of conclusion added submitted for internal review.
0.6	25/07/2024	Yannick Roelvink Emna Amri David Puron	Integrated comments from reviewers regarding Chapters 1, 2, 3.2, 3.3 and 4.
0.7	26/07/2024	Yannick Roelvink Shaza Zeitouni Tymoteusz Burak	Integrated comments from reviewers regarding Chapters 3.1, and 3.5
0.8	29/07/2024	Yannick Roelvink	Final version for QA.
0.9	30/07/2024	Juan Alonso	Quality Assessment.
1.0	31/07/2024	Hristo Koshutanski	Final version submitted.

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Yannick Roelvink (CYSEC)	29/07/2024
Quality manager	Juan Alonso (ATOS)	30/07/2024
Project Coordinator	Hristo Koshutanski (ATOS)	31/07/2024

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
API	Application Programming Interface
D5.2	Deliverable number 2 belonging to WP5
EC	European Commission
FCU	Flight Control Unit
FPGA	Field Programmable Gate Arrays
GNN	Graph Neural Network
HW	Hardware
IoT	Internet of Things
IPC	Inter-Process Communication
MFA	Multi-Factor Authentication
MitM	Man-in-the-Middle (cybersecurity attack)
ML	Machine Learning
MPU	Memory Protection Unit
MS5	5th Project Milestone
mTLS	Mutual Transport Layer Security
OS	Operating System
PG	Perimeter Guard
PUF	Physical Unclonable Function
RA	Remote Attestation
REE	Rich Execution Environment
SoC	System on Chip
SPMP	S-mode Physical Memory Protection
SRAM	Static Random-Access Memory
SW	Software
TA	Trusted Application
TEE	Trusted Execution Environment
UAV	Unmanned Aerial Vehicle
UC	Use Case
vFPGA	Virtual Field Programmable Gate Arrays
VM	Virtual Machine
WP	Work Package

Executive Summary

Deliverable D5.2 details the effort performed regarding the integration of the CROSSCON Stack. This includes (1) an update of the development status of the CROSSCON Stack components and trusted services, on which the integration, testing and validation activities of WP5 depend, as well as (2) a detailed description of the use case implementation prototypes that will be integrated into the testbed defined in T5.1, as preparation of the testing and validation activities of T5.3 and T5.4.

Where available, the development status of the CROSSCON Stack is illustrated using a description of a component's or trusted service's technical functionality. In addition, for each of the Use Case prototypes, a detailed description of their implementation architecture and their required components, trusted services and interfaces of the CROSSCON stack will be provided. Furthermore, the deliverable will provide an initial timeline of the integration of each of the use cases into the testbed environment.

Deliverable D5.2 contributes to the accomplishment of milestone MS5 "First version of integrated CROSSCON Stack and extension primitives, and first version of business model and market proposition".

1 Introduction

1.1 Purpose of the document

This document provides a demonstration of the current operational functionality of the CROSSCON Stack. This is achieved by listing and demonstrating the development status of the Stack's components and trusted services required for the integration, testing and validation activities of WP5, as well as a detailed overview of each of the use case implementation prototypes and their required security features. In addition, based on the status of the CROSSCON Stack, a timeline of the integration of each of the use case prototypes into the testbed is provided. This is crucial, as these use case prototypes will be used as a first reference for the activities of T5.3 and T5.4, related to the security testing of the use cases and CROSSCON Stack validation, respectively.

1.2 Relation to other project work

This document aims to summarize the overall status of the CROSSCON Stack. As such, it derives the use case descriptions and operational scenarios, used to define the use case implementation prototypes, from D1.4[1]. In addition, the implementation prototypes presented in this document will be validated according to the validation criteria provided in D1.6.

The CROSSCON Stack components, trusted services and extension primitives presented in this document are based on the definitions of the CROSSCON Stack as provided in D3.1[4] and D4.1[6], respectively. In addition, some of the technical demonstration descriptions provided in this document are based on the development advancements presented in D3.2[5] and D4.2[7].

The outputs of this document are necessary inputs for the security testing and validation of the use case prototypes, as defined in D5.3 & D5.4.

1.3 Structure of the document

This document is structured in 3 major chapters. After this introductory chapter, Chapter 2 will provide an overview of the development progress of the CROSSCON Stack, detailing the status of each of its components and trusted services, as well as providing a demonstration of their functionality where available. Afterwards, Chapter 3 will provide a detailed overview of the use case integration prototypes, including their required security features, and an associated integration plan, split into sections for each of the use cases. Lastly, Chapter 4 will summarize the key conclusions of the work presented in this document.

2 CROSSCON Stack Development Progress

This chapter provides an overview of the current development status of the CROSSCON Stack. In section 2.1, we present the status of the core components needed to realize the different instantiations of the CROSSCON, while in section 2.2, we describe the status of the high-level security function provided by the trusted services.

Figure 1, taken directly from D2.1[3], recalls the detailed architecture of the CROSSCON stack and its components.

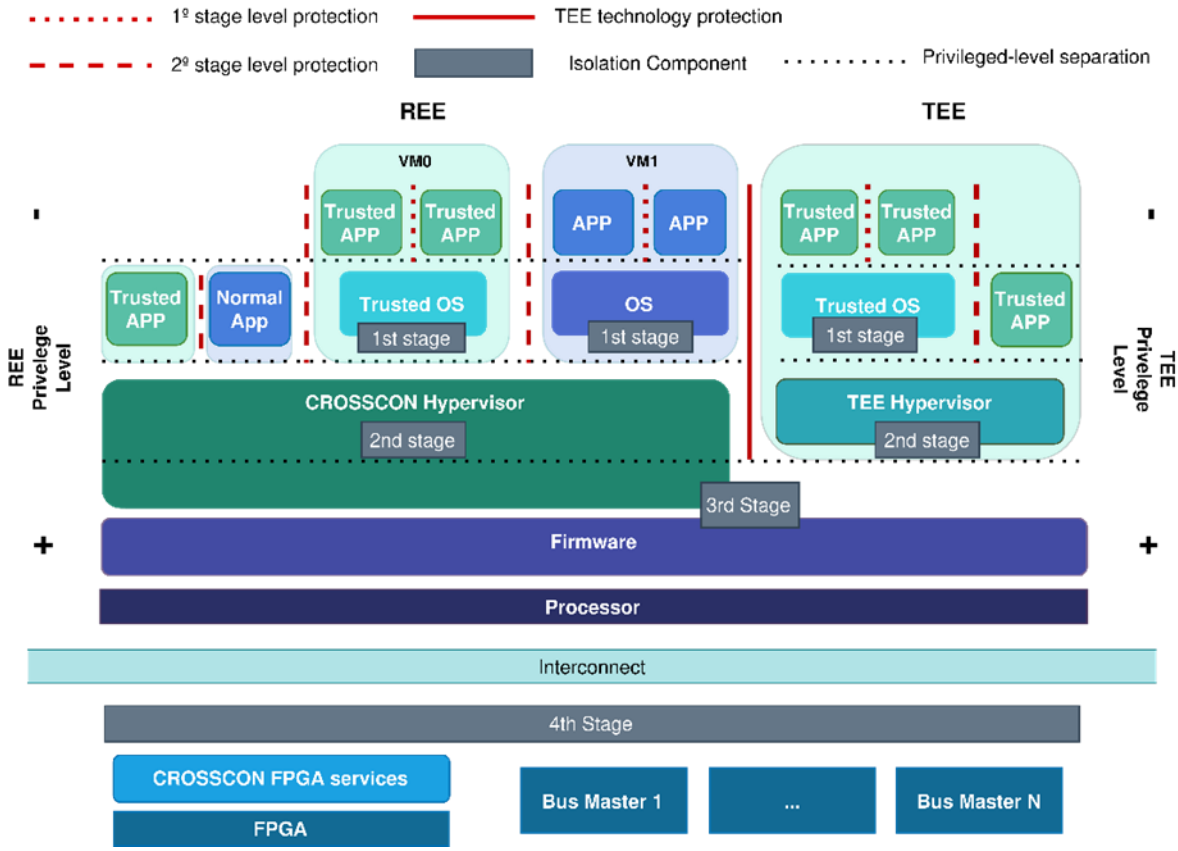

Figure 1: Detailed CROSSCON Stack architecture[3]

2.1 Status of CROSSCON Stack Components

Table 1 provides an overview of each of the CROSSCON Stack components' development progress, including a description of the component, its current integrability into the use case setups and a demonstration of its current technical capabilities, where available. For a more detailed description regarding each of the components, please refer to sections 3.2, 3.4 and 3.5 of D3.1[4]. For an overview of the code used for the technical demonstrations, please refer to D3.2[5] and D4.2[7].

Table 1: Progress overview of CROSSCON Stack components

Component	Description	Current Integration Status	Technical Demonstration
CROSSCON Hypervisor	The CROSSCON Hypervisor is a static partitioning Hypervisor that aims to provide strong isolation and real-time guarantees. It is based on the Bao[8] hypervisor, but implements additional features such as (1) dynamic VM creation & management, (2) per-VM TEE service support and (3) multiple Virtual Machine Manager (VMM) support.	Although the per-VM TEE service support has been integrated and tested, the other two features are still under development. In addition, the CROSSCON Hypervisor is currently only compatible with QEMU[9], an open-source emulator used to virtualize hardware platforms. While QEMU allows for the testing of the Hypervisor and its components prior to integration, work is currently being performed to ensure compatibility with several of the use case hardware platforms required for integration.	Using the QEMU platform, the CROSSCON Hypervisor is able to demonstrate the ability to provide per-VM TEE services, isolating the TEE by running multiple TEE VMs instead of a single TEE instance in the normal world.
CROSSCON bare metal TEE	In order to still provide security for low-end systems that are not compatible with the aforementioned CROSSCON Hypervisor, CROSSCON proposes a software-based bare-metal TEE, to allow bare-metal devices to interact securely with the rest of the CROSSCON Stack. The TEE will ensure the basic security primitives such as memory isolation, privilege separation and cross-domain communication.	The bare-metal TEE has been integrated and validated on both non-MPU and MPU devices. Although both solutions provide a similar set of security guarantees to deployed applications, they do so using a slightly different isolation mechanism.	Two prototypes of the bare-metal TEE are currently available: one where the MPU version is implemented on an ARMv7-M architecture and another where the non-MPU version is deployed on a MSP430 architecture, both illustrating the required basic security primitives.

Component	Description	Current Integration Status	Technical Demonstration
	Two distinct versions of the bare-metal TEE are being developed, one for bare-metal devices that do not have a Memory Protection Unit (MPU), and one for devices that do.		
CROSSCON TEE Toolchain	In order to manage the development and implementation of TEEs and TAs on the CROSSCON Stack, a toolchain is being developed that aims to support these activities. More specifically, the toolchain will be used to perform two critical tasks: ● **Secure IoT firmware updates:** Based on the IETF proposed standard as detailed in RFC 9019 (SUIT), it allows for a secure mechanism to perform firmware updates on IoT devices[12]. ● **Secure cross-compilation of TAs for different TEEs:** The toolchain will include a secure compiler, used to guarantee the memory safety of the TAs deployed on the TEEs within the CROSSCON Stack.	The advancements of the TEE Toolchain can be grouped into three categories: **Secure Update:** The design for the secure firmware update feature is done, and the development for a first implementation is still ongoing. More information regarding this secure update mechanism can be found in D5.1 **Integration into DevSecOps:** The toolchain will be integrated into the DevSecOps pipeline that will be used for the integration of the TAs onto the CROSSCON Hypervisor within the testbed environment of T5.1. This integration is currently still ongoing. **Secure TA cross-compilation:** The definition of a secure compiler, which takes code written from a simplified version of C to a memory safe version of the same code, has been performed. The implementation of this compiler, as well as the extension	Not available

Component	Description	Current Integration Status	Technical Demonstration
		to lower programming languages, is currently ongoing.	
CROSSCON SoC	CROSSCON SoC[1] is a system-on-chip (SoC) design, developed as part of the CROSSCON project, that provides a secure RISC-V execution environment for mixed-criticality IoT devices that require strong security guarantees, flexibility, small code size and low power consumption. The CROSSCON SoC guarantees strong software isolation with the ability to share hardware modules connected to the SoC interconnect between TEEs without compromising isolation guarantees.	A first implementation of the CROSSCON SoC has been developed, designed around the BA51-H core, which also integrates an implementation of unified (2-stage) S-mode Physical Memory Protection (SPMP) RISC-V extension. In addition, although a basic deployment is already available, work is still being performed to fully port the CROSSCON Hypervisor on top of the BA51-H CROSSCON SoC, in order to provide a full secure Stack implementation.	The current implementation of the CROSSCON SoC has successfully been integrated with the BA51-H core, and provides a Perimeter guard demonstrator that allows one to share the SRAM module between isolated domains / VMs.
CROSSCON Perimeter Guard	A Perimeter Guard (PG) module is a hardware (HW) module that can be used to share HW modules between isolated domains, without compromising the isolation itself. PG achieves this by (1) controlling which SoC interconnect master / application can access the protected HW module and by (2) managing the HW module's state during domain switching (i.e. when the HW access get handed over from one application to another), to ensure no unwanted information flows occur.	A prototype implementation, which supports a lock-release with time-sharing operation mode, is available as part of the current implementation of the CROSSCON SoC. The development of additional modes of operation and arbitration is currently ongoing, and additional tests are being performed to extend the range of HW modules the PG can be applied to.	The current implementation of the PG module has successfully been integrated into the CROSSCON SoC, and can be used to limit access to the Static Random-Access Memory (SRAM) HW module.

[1] Description of CROSSCON SoC (chapter 2.2 of D4.2[7])

2.2 Status of CROSSCON Stack Trusted Services

Table 2 provides an overview of each of the CROSSCON Stack trusted services' development progress, including a description of the service, its current integrability into the use case setups and a demonstration of its current technical capabilities, where available. For a more detailed description regarding each of the trusted services, please refer to section 3.3 of D3.1[4]. For an overview of the code used for the technical demonstrations, please refer to D3.2[5] and D4.2[7].

Table 2: Progress overview of CROSSCON Stack trusted services

Trusted Service	Description	Current Integration Status	Technical Demonstration
PUF-based authentication	CROSSCON aims to innovate and integrate a secure, efficient, lightweight and scalable PUF-based authentication scheme, facilitating resource-constraint embedded devices to authenticate themselves between each other, based on their inherent hardware variations. During the project, three main implementation methods have been identified: **ZK-PUF:** PUF-based authentication via zero-knowledge proofs, **PAVOC:** PUF-based authentication via one-way chains, and **PAWOS:** PUF-based authentication via one-time signatures.	All three of the proposed PUF-based authentication schemes have been implemented and tested on the NXP LPC55S6x hardware board, as part of the integration efforts for UC1. This implementation shows the compatibility of the trusted service with a virtualization-less deployment of the CROSSCON Stack. We are currently working on integrating the PUF-based authentication mechanisms into the CROSSCON Stack as a Trusted Application (TA)	The three implementations on the LPC55S6x hardware board are able to demonstrate both of the operational phases of PUF-based authentications **Enrollment:** The systems correctly generated the public data from the PUF responses of the IoT device. **Authentication:** The verification device is able to correctly verify the authenticity of the IoT device using its PUF responses and the public data.
Context-based authentication	As an additional authentication method, a context-based authentication scheme is being developed, which utilizes the Wi-Fi landscape around a device to provide it with	A first implementation of the context-based authentication has been implemented on the RPi 4B board, which was able to collect samples of the metadata of the Wi-Fi	Not available

	a unique fingerprint. This fingerprint can be used to authenticate the device to an authoritative party, based on Siamese networking and machine learning to validate the similarity between a submitted and enrolled fingerprint of the networking landscape of the device.	landscape required for fingerprint generation. Tests are currently ongoing to expand its compatibility with other hardware platforms. In addition, first experiments with the Siamese networking have been performed, but further testing is required before it can be integrated and validated as a trusted service.	
Secure FPGA provisioning	The secure FPGA provisioning service of the CROSSCON Stack will allow its clients to deploy and operate remote shared FPGA hardware platforms, while at the same time ensuring (1) the confidentiality of the proprietary FPGA workloads and designs, and (2) protection against malicious workloads from other users and/or malicious logic insertions into the design itself. It achieves this by provided two main sub-services: ● **FPGA Configuration service:** Which is responsible for securely deploying and configuring a user's FPGA design onto a remote FPGA hardware module. ● **FPGA Bitstream Scanning service:** Which is responsible for ensuring the user's designs are free from	Although the secure FPGA provisioning service is still in development, an important step toward has been made regarding the design and implementation of the FPGA shell in the FPGA fabric. This shell will be responsible for configuring the rest of the setup at runtime.	The current prototype of the secure FPGA provisioning service is able to reconfigure two virtual FPGAs at runtime with different accelerators through the internal configuration port.

	malicious circuits before being deployed on the FPGA device.		
Behavioral-based network anomaly detection	A network anomaly detection service is being developed, which will monitor the networking activities of an embedded device, and detect any deviations from its regular behavior. These deviations include, but are not limited to, suspicious port numbers being exposed by the device, or suspicious communication and/or protocols being used with other devices or IP addresses.	The behavioral-based network anomaly detection service implementation of the CROSSCON Stack is currently still under development. Once the service is ready, it will be integrated into the CROSSCON Stack as a Trusted Application (TA).	Not available
Control flow integrity	CROSSCON aims to extend protection against control flow attacks, i.e. attacks aimed at diverting the sequence of executed instructions of an application, to IoT devices that either lack or are not compatible with existing hardware-specific solutions. Two control flow integrity service implementations, protecting either the backwards edge, ensuring that each function returns to the point in the code where it was called, and/or the forwards edge of the control flow, ensuring the destination address of the execution is part of a set of allowed destinations, are developed: **Flashadow:** backwards edge control flow protection for the CROSSCON	A first implementation of both the Flashadow and uIPS control flow mechanisms have been integrated onto the CROSSCON bare metal TEE for non-MPU and MPU hardware modules, respectively. More testing is currently ongoing to further validate these implementations.	The current implementation for both MPU and non-MPU devices are able to demonstrate the ability to protect the forward and backward edges (where applicable) of the control flow, detecting unsafe deviations from the original sequence of operations and halting the execution in such cases.

	non-MPU bare metal TEE (designed for Class 0 devices, as defined in D1.5[2]) **uIPS:** backwards and forwards edge control flow protection for the CROSSCON MPU bare metal TEE (designed for Class 1 devices, as defined in D1.5[2])		
Remote attestation	Remote attestation is an existing security mechanism that allows a device to verify the integrity and authenticity of another remote device, ensuring it has not been compromised or tampered with. CROSSCON aims to enhance the capabilities of regular remote attestation by providing a service that allows users to easily configure and call a remote attestation of one of the VMs running upon its Stack.	The remote attestation implementation of the CROSSCON Stack is currently still under development. Once the service is ready, it will be integrated into the CROSSCON Stack as a Trusted Application (TA).	Not available
ML-based control flow attestation	This service allows the user to utilize remote attestation to verify the integrity of the control flow of a vulnerable and/or security-relevant application, using Unsupervised Graph Neural Networks (GNNs) to detect any deviations from regular operational sequences.	The implementation of the ML-based runtime attestation of the CROSSCON Stack is currently still under development. Once the service is ready, it will be integrated into the CROSSCON SoC as a peripheral.	Not available

3 Use Case Integration Prototypes

Within Chapter 3, a detailed description of each of the use case implementation prototypes will be provided, split into sections for each use case. In addition to the architecture of the use case integration prototypes, this chapter will highlight the components and trusted services each of the use cases will leverage for increased security, as well as a first integration timeline.

It is important to note that the prototype architectures and integration timelines are subject to changes, depending on the input and development timeline of the CROSSCON Stack, as well as its components and services.

3.1 UC 1 - Device Multi-Factor Authentication

3.1.1 Prototype Architecture

The main goal of UC 1 is to provide a Multi-Factor Authentication (MFA) implementation for lightweight and resource-constraint IoT devices, in order to protect their users from Man-in-the-Middle (MitM) attacks, amongst others. It achieves this by implementing a basic first authentication factor based on a mutual Transport Layer Security (mTLS) TA implementation. For the secondary authentication, the UC will leverage either the PUF-based or Context-based authentication schemes developed as part of the trusted services of the CROSSCON Stack, depending on their availability on the underlying hardware (see Table 1 of D3.1[4] for an overview of the features of each of the hardware platforms considered for CROSSCON).

More specifically, UC 1 aims to provide a MFA implementation for two distinct operational scenarios:

3.1.1.1 Scenario 1: One-way authentication between a low-end and a high-end device

Within the first operational scenario, UC 1 proposes the following implementation to provide a one-way authentication mechanism between the resource-constrained IoT devices and gateways:

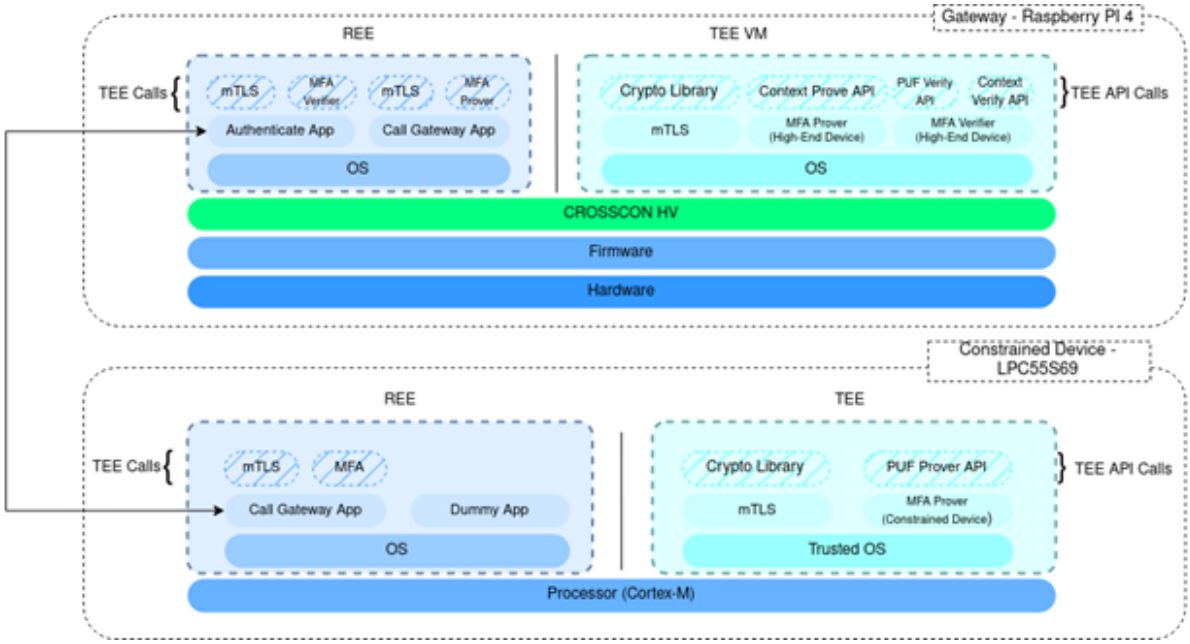

Figure 2: MFA implementation prototype for low-end to high-end device authentication

Utilizing the architecture as defined in Figure 2, the following workflow will be implemented:

0. The constrained IoT device, based on a NXP LPC55S6x, enrolls its PUF responses to a public database, as described in Section 3.3.1 of D3.1[4], prior to its required authentication.
1. The constrained IoT device sends a mTLS request to the gateway, itself based on a Raspberry Pi 4B.
2. Once the gateway receives the request from the constrained device, it verifies the provided certificate.
    a. If the certificate is valid, a secure encrypted channel is established between the constrained device and the gateway. The gateway then sends an authentication response to the constrained device, prompting for a second authentication factor.
    b. If the certificate provided by the constrained device is invalid, the gateway sends a negative authentication response and does not prompt for the second authentication factor.
3. In case of a successful first authentication, the constrained IoT generates a secondary authentication using its PUF-based MFA implementation, and forwards this to the gateway.
4. The gateway verifies the received PUF-based authentication using the aforementioned public data.
    a. If the second factor authentication fails, the gateway sends a negative authentication response back to the IoT device.
    b. If the second factor is valid, the gateway sends a positive authentication response.

3.1.1.2 Scenario 2: Mutual authentication between two high-end devices

Where the first scenario focuses on the one-way authentication of a low-end to a high-end device, the second scenario explores the possibility to provide a more complex, mutual authentication between two high-end devices. The proposed implementation architecture can be found in Figure 3:

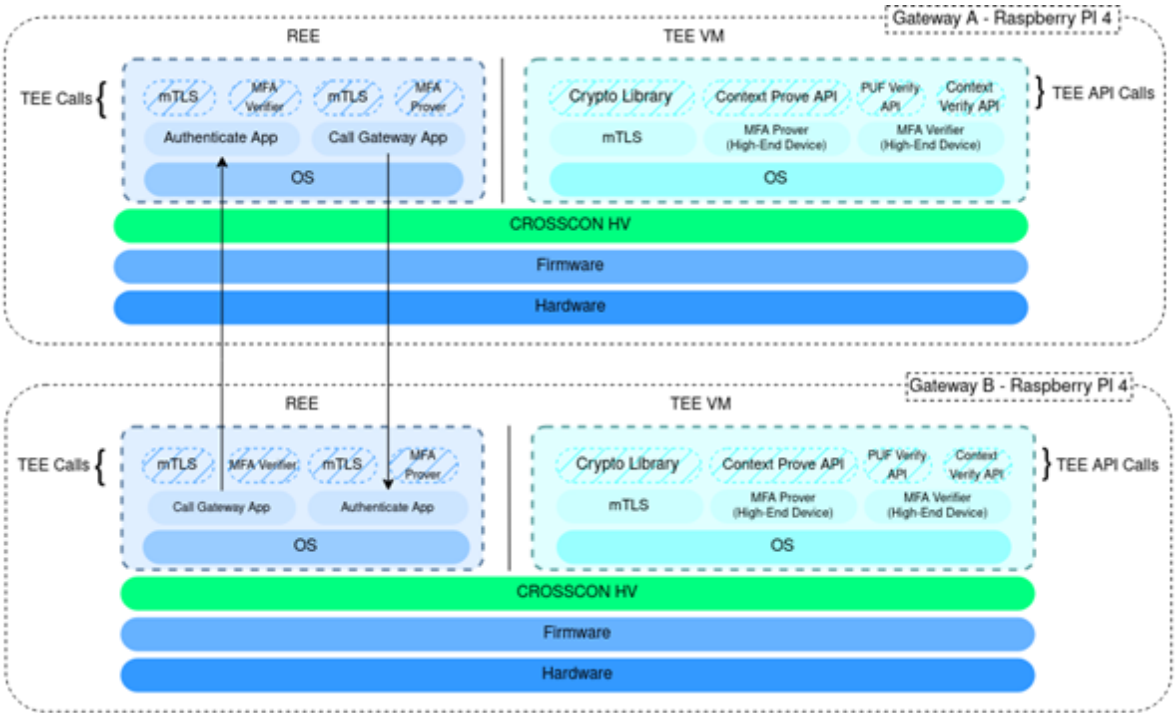

Figure 3: MFA implementation prototype for mutual authentication between two high-end devices

As the high-end gateway devices are more powerful and have extended capabilities compared to the low-end devices of the first scenario, the MFA will be based on context-based, rather than PUF-based authentication, as described in the following workflow:

0. Both Gateway A and Gateway B, each based on a Raspberry Pi 4B hardware module, have enrolled their Wi-Fi landscape fingerprints with each other, in order to be able to validate each other's Context-based MFA.
1. Gateway A sends a mTLS request to Gateway B.
2. Once Gateway B receives the authentication request, it verifies the provided certificate.
    a. If the certificate is valid, a secure encrypted channel is established between the two gateways. Gateway B then sends an authentication response to Gateway A, prompting for a second authentication factor.
    b. If the certificate provided by Gateway A is invalid, Gateway B sends a negative authentication response and does not prompt for the second authentication factor.
3. In case of a successful first authentication, the Gateway A generates a secondary authentication using its Context-based MFA implementation, and forwards this to Gateway B.
4. Gateway B verifies the Context-based authentication using the Siamese network implementation of the MFA TA.
    a. If the second factor is invalid, Gateway B sends a negative authentication response.
    b. If the second factor is valid, the Gateway B sends a positive authentication response, and provides its own secondary Context-based authentication factor to Gateway A.
5. Gateway A verifies the received Context-based authentication from Gateway B using its own Siamese network.
    a. If the second factor is invalid, Gateway A sends a negative authentication response.
    b. If the second factor is valid, Gateway A returns a positive authentication response.

3.1.2 Required Security Features

The integration and operation of the two architectures described before will leverage the following security features, obtained from the components and trusted services of the CROSSCON Stack:

- **CROSSCON Hypervisor support on hardware platforms:** The implementations of both the first and second operational scenario of UC 1 are designed to operate using the CROSSCON Hypervisor on top of either a NXP LPC55S6x or Raspberry Pi 4B hardware platform. However, as described in Section 2.1, the CROSSCON Hypervisor is currently only compatible with QEMU, and efforts to ensure its compatibility with the required hardware platforms are ongoing.
- **PUF-Based Authentication Prover API for Low-End devices:** An API will have to be developed for the PUF-based authentication implementation on low-end devices, such that they can generate PUF responses whenever a MFA is required.
- **PUF-Based Authentication Verifier API for High-End devices:** An API will have to be developed for the PUF-based authentication implementation on high-end devices, such that they can validate the PUF-based authentication of low-end devices, when requested.
- **Context-Based Authentication Prover & Verifier API for High-End devices:** An API will have to be developed for the Context-based authentication implementation on high-end devices, such that they can both (1) generate their Wi-Fi landscape fingerprint, and (2) validate the Wi-Fi landscape fingerprint of another high-end device, when requested.

3.1.3 Integration Timeline

Considering the proposed implementation architectures and operational scenarios, as well as the ongoing development of the CROSSCON Stack, the following integration timeline is predicted for the use case prototype:

Table 3: Integration timeline for UC 1

Integration Activity	2024						2025									
	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10
Integration of the CROSSCON Stack on Raspberry Pi 4B																
Integration of the CROSSCON authentication implementations																
Integration of the MFA application																
Testing and validation activities																
Preparation of final demonstrator																

3.2 UC 2 - Firmware Updates of IoT Devices

3.2.1 Prototype Architecture

The integration prototype architecture for the UC2, focussing on a secure Over-the-Air (OTA) firmware update implementation lead by BIOT, is the following:

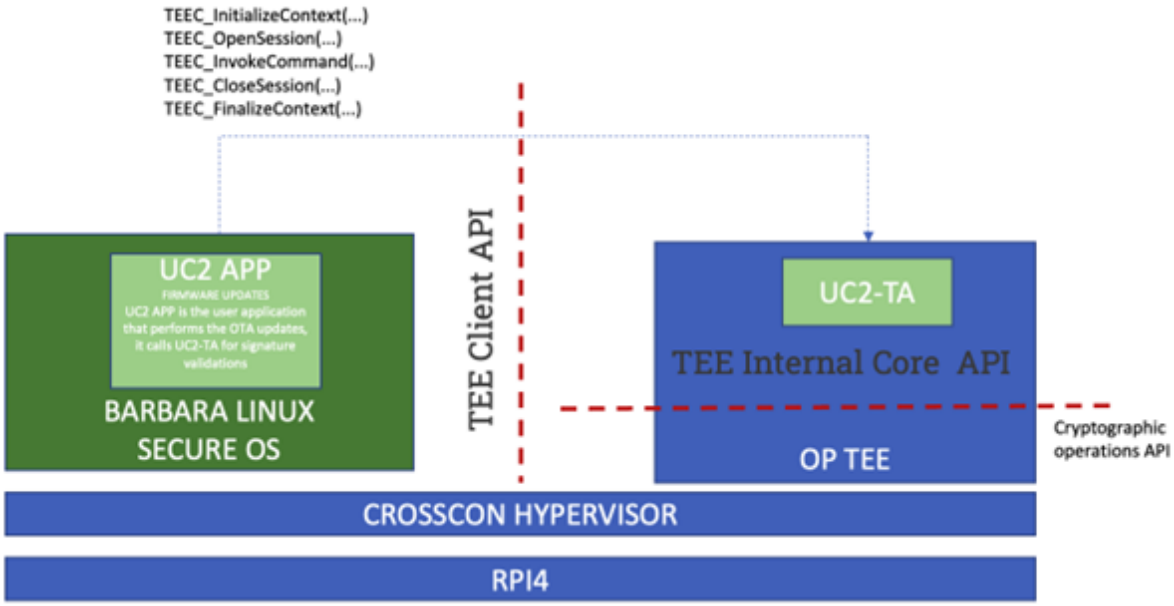

Figure 4: Secure firmware update implementation prototype

As depicted in Figure 4, Barbara Secure OS will be deployed inside a VM on top of the CROSSCON Hypervisor, operating on a Raspberry Pi 4B hardware platform. Within the Barbara VM, the UC2 operational application (UC2 APP) will be deployed, which will be responsible for performing the OTA firmware updates. It leverages an external update server[2] to provide the required materials to execute the OTA updating operations.

The UC2 application will utilize a specialized Trusted Application, called UC2-TA, responsible for validating the signature of the downloaded firmware file, using the TEE client API[10]. This TA is deployed on OP-TEE, a TEE instance operating on top of the CROSSCON Hypervisor. Afterwards, the UC2-TA application will utilize the TEE internal Core API[11], in order to perform its critical cryptographic functions.

The operational workflow of UC 2 will be the following:

0. An Operating System update image will be generated and stored in the dedicated update server.
1. The UC2 OTA will ask the CROSSCON API for a unique ID. This ID will be stored on the device using a secure storage area, provided by the OP-TEE implementation operating on top of the CROSSCON Hypervisor.
2. The CROSSCON Stack will be used to set up the secure communication channel between the IoT device and the update server, and the OS update image will be downloaded as an encrypted binary file.
3. The UC2-TA application will verify the signature of the encrypted binary file, using the TEE cryptographic operations.
4. If the signature is valid, the UC2-TA application will decrypt the update image file using the TEE cryptography operations provided by the TEE API.

[2] This server will be deployed and operated independently of the aforementioned implementation prototype, not requiring integration with the CROSSCON Stack, and has therefore been omitted from the detailed architecture of Figure 4.

3.2.2 Required Security Features

The integration and operation of the two architectures described before will leverage the following security features, obtained from the components and trusted services of the CROSSCON Stack:

- **CROSSCON Hypervisor support on RPI4:** The architecture of UC 2 is designed to operate using the CROSSCON Hypervisor on top of a Raspberry Pi 4B hardware platform. However, as described in Section 2.1, the CROSSCON Hypervisor is currently only compatible with QEMU, and efforts to ensure its compatibility with the Raspberry Pi 4B hardware platform are ongoing.
- **CROSSCON TEE implementation:** The UC2-TA application requires to be deployed on a OP-TEE implementation on top of the CROSSCON Hypervisor, as the TA will leverage the TEE implementation to (1) provide the secure storage capabilities, (2) allow for the generation of a unique identifier and (3) perform the cryptographic operations required for the use case.
- **Unique identifier (ID) provisioning:** The OTA firmware update application requires a unique device ID to be provisioned by the CROSSCON Stack. The implementation of this service is not considered as a trusted service, but will instead leverage the secure storage capabilities of the CROSSCON TEE implementation to generate and store the unique ID.
- **Secure storage capability:** The CROSSCON Stack will provide the ability to allocate secure storage to store sensitive data (e.g. cryptographic keys, unique ID, etc.) only accessible by allowed applications.
- **Secure Communication Channel:** To ensure the authenticity, integrity and confidentiality of the communication between the IoT device and the firmware update server, a method to establish a secure communication channel will have to be available.

3.2.3 Integration Timeline

At first, the integration of the prototype into the testbed will be performed, which will serve as a first evaluation of the CROSSCON Stack, and provide feedback to the developers of WP2, 3 and 4. The integration schedule of this first implementation is presented in Table 4, and has been divided in three integration phases:

1. Development of the emulated environment setup, where we can begin creating and running the Hypervisor, the TEE and Barbara OS on top of CROSSCON Hypervisor.
2. Development of UC2-TA, and its integration within the CROSSCON TEE environment
3. Development of the UC2 OTA update application, its integration within the Barbara OS VM and the establishment of the communication between the UC2 OTA update application and the UC2-TA.

After the integration, a continuous testing and integration framework will be implemented, in order to continue validating the advancements made, and to provide feedback to the development of the CROSCONN Stack.

Table 4: Integration timeline for UC 2

Integration Activity	2024						2025									
	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10
Development of the emulated environment setup																
Development of UC2-TA																
Development of the UC2 OTA update application																
Continuous Testing and validation activities																
Preparation of final demonstrator																

3.3 UC 3 – Commissioning and Decommissioning of IoT Devices

3.3.1 Prototype Architecture

The integration prototype architecture for the UC 3, focussing on a secure framework for commissioning and decommissioning of IoT devices lead by BIOT, is the following:

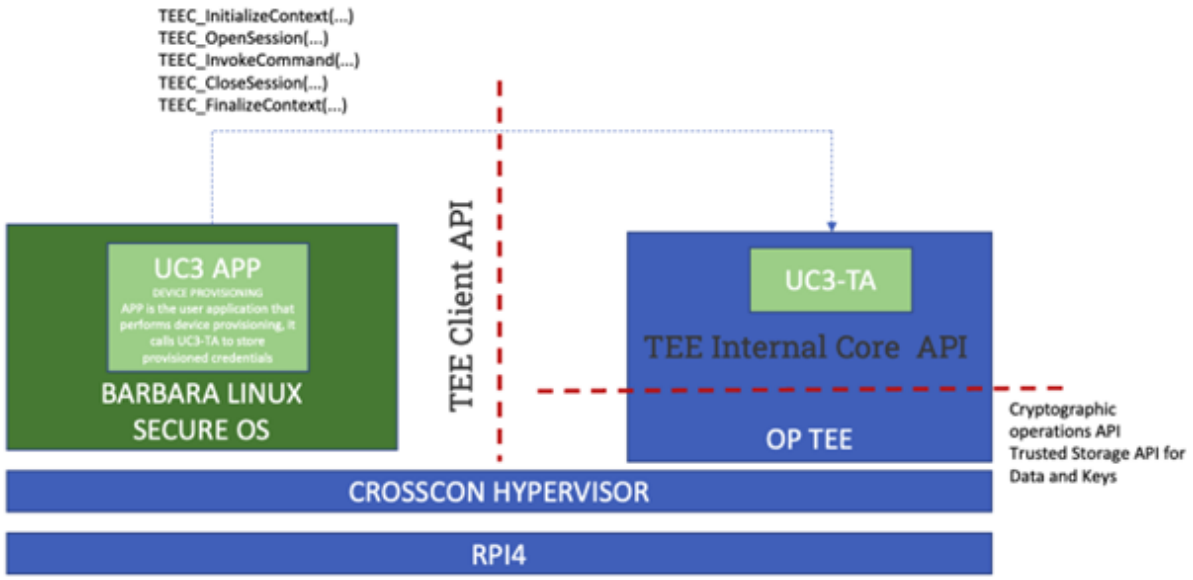

Figure 5: Secure (de)commission of IoT devices implementation prototype

As depicted in Figure 5, the implementation of UC 3 follows almost exactly the same architecture as that of UC 2: A Barbara Secure OS will be deployed inside a VM on top of the CROSSCON Hypervisor, operating on a Raspberry Pi 4B hardware platform. Within the Barbara VM, the UC 3 operational application (UC3 APP) will be deployed, which will be responsible for performing the (de)commissioning operations of the IoT device. It leverages an external provisioning server[3] to provide the required materials to execute these (de)commissioning operations.

The UC3 application will utilize a specialized Trusted Application, called UC3-TA, responsible for executing the required security-related operations, using the TEE client API[10]. This TA is deployed on OP-TEE, a TEE instance operating on top of the CROSSCON Hypervisor. Afterwards, the UC3-TA application will utilize the TEE internal Core API[11], in order to perform its critical cryptographic functions.

The operational scenario of UC 3 will be the following:

0. Upon first boot, the UC3-TA application will generate a unique ID and store it using a secure storage area provided by the OP-TEE implementation operating on top of the CROSSCON Hypervisor. It will also generate a generic certificate and store it in the same secure storage.
1. Next, the UC3 application will login to the dedicated provisioning server using the aforementioned generic certificate and unique ID, provided by the UC3-TA.
2. The provisioning server verifies the credentials provided by the IoT device, and upon successful authentication, sends an official production certificate back to the IoT device. This certificate is then stored in the aforementioned secure storage area, replacing the generic certificate used during the enrollment process.
3. After enrollment, the IoT device can be operated, utilizing the production certificate from step 3 to officially authenticate itself to third-party applications if required.

[3] This server will be deployed and operated independently of the aforementioned implementation prototype, not requiring integration with the CROSSCON Stack, and has therefore been omitted from the detailed architecture of Figure 5.

3.3.2 Required Security Features

The integration and operation of the aforementioned architecture depend on the advancement of the following components and trusted services of the CROSSCON Stack:

- **CROSSCON Hypervisor support on RPI4:** The architecture of UC 3 is designed to operate using the CROSSCON Hypervisor on top of a Raspberry Pi 4B hardware platform. However, as described in Section 2.1, the CROSSCON Hypervisor is currently only compatible with QEMU, and efforts to ensure its compatibility with the Raspberry Pi 4B hardware platform are ongoing.
- **CROSSCON TEE implementation:** The UC3-TA application requires to be deployed on a OP-TEE implementation on top of the CROSSCON Hypervisor, as the TA will leverage the TEE implementation to (1) provide the secure storage capabilities, (2) allow for the generation of a unique identifier and (3) perform the cryptographic operations required for the use case.
- **Unique identifier (ID) provisioning:** The (de)commissioning application requires a unique device ID to be provisioned by the CROSSCON Stack. The implementation of this service is not considered as a trusted service, but will instead leverage the secure storage capabilities of the CROSSCON TEE implementation to generate and store the unique ID.
- **Secure storage capability:** The CROSSCON Stack will provide the ability to allocate secure storage to store sensitive data (e.g. cryptographic keys, unique ID, etc.) only accessible by allowed applications.
- **Secure Communication Channel:** To ensure the authenticity, integrity and confidentiality of the communication between the IoT device and the provisioning server, a method to establish a secure communication channel will have to be available.

3.3.3 Integration Timeline

The integration schedule of this first implementation is presented in Table 5, and has been divided in three integration phases, similar to the one introduced for UC 2:

1. Development of the emulated environment setup, where we can begin creating and running the Hypervisor, the TEE and Barbara OS on top of CROSSCON Hypervisor (which follows the same timeline as the UC2 integration timeline).
2. Development of UC3-TA, and its integration within the CROSSCON TEE environment
3. Development of the UC3 OTA update application, its integration within the Barbara OS VM and the establishment of the communication between the UC2 OTA update application and the U2-TA.

After the integration, a continuous testing and integration framework will be implemented, in order to continue validating the advancements made, and to provide feedback to the development of the CROSCONN Stack.

Table 5: Integration timeline for UC 3

Integration Activity	2024						2025									
	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10
Development of the emulated environment setup																
Development of UC3-TA																
Development of the UC3 (de)commissioning application																
Continuous Testing and validation activities																
Preparation of final demonstrator																

3.4 UC 4 - Remote Attestation for Identification and Integrity Validation of Agricultural UAVs

3.4.1 Prototype Architecture

Within UAVs, the Flight Control Unit (FCU), is responsible for the operations of the drone's movements, executing either the commands received from the UAV pilot's controller, or acting as an auto-pilot for preloaded flight plans. In addition, it is often the sole entrance point for the UAV flight peripheral sensor data (e.g. GPS, air speed, etc), used by the FCU to maintain flight, as well as the only communication point to the flight controllers. Hence, it is a critical component for the operational safety of the UAV, and therefore crucial to secure.

However, in almost all cases the FCU is either a separate hardware or software module that is not developed or maintained by the UAV owner, and as such cannot be inherently trusted. In addition, the FCU often operates alongside a business payload, which executes the commercial applications of the agricultural UAV (e.g. spraying, monitoring, etc). As these commercial operations required input from the drone's flight peripherals, another main task of the FCU is to share any required data with the business payload, meaning that any tampering performed on the FCU will have a direct impact on the commercial implementations of the UAV.

For these reasons, UC 4 aims to ensure the operational safety and security of the UAV by ensuring the FCU operates nominally. It achieves this by implementing the following operational scenario, leveraging Remote Attestation (RA) as a means to validate the status of the FCU:

1. The business payload requests the sensor data from the FCU during operations, but before utilizing its values for the business logic, it checks them against a list of critical values preinstalled by the UAV owner (e.g. max legal airspeed, gps restrictions, etc).
    a. If these values are within the specified critical range, the operations are nominal and no further actions are taken.
    b. However, if the values exceed the critical range, the business payload will trigger the RA service to validate the integrity of the FCU implementation.
2. If the critical range is exceeded, the RA service will perform an attestation of the status of the FCU and send the attestation report to a RA verifier, located on a centralized server, via a secure communication channel. The RA verifier will check whether the FCU has been compromised, and will report back to the RA service on the UAV regarding its conclusion.
    a. If the attestation fails, i.e. the FCU has been compromised, an evasive action needs to be taken to ensure the operational safety of the UAV. As the exact operations that could be considered in this scenario depend on the architecture of the drone, its components and/or the criticality of the mission, UC 4 does not impose any evasive actions that need to be taken. Instead, it allows the UAV owner to pre-program a system to execute an evasive action in case a failed attestation occurs.
    b. If the attestation does pass, i.e. the FCU has not been compromised with, an additional check is performed to locate the reason for the previously measured out-of-range values of the sensor data.
3. If the RA service validates the integrity of the FCU, the business logic will request an additional set of sensor data results.
    a. If these additional sensor data points are also out of range, the FCU is instructed to correct its operations to return to a safe state (e.g. reducing its flight velocity, lowering its altitude, etc). Example scenarios where such unsafe conditions could occur include, but are not limited to, dangerous manual flying from the UAV pilot, or an erroneous UAV remote control.
    b. If the new sensor data sample is showing data that is no longer out of range, the output of the sensor in question is no longer trustworthy (e.g. it is being jammed or is defective). Depending on the hardware capabilities of the UAV, evasive actions can be

taken, to ensure the faulty sensor data is not interfering with the operations of the UAV. For example, if there are redundant alternatives to the corrupted sensor, the FCU can be instructed to ignore its output completely. Once again, UC 4 does not impose any evasive actions, but instead allows the UAV owner to pre-program a system to execute an evasive action in line with their design and requirements.

4. Of course, an attacker could bypass the attestation of step 2 if it tampers with the FCU and masks its efforts by altering the output data to the business peripheral, making it seem as if the sensor values are within the critical range, even when they are not. To combat this, the UAV owner will be able to schedule periodic attestations from within the business payload application, which would trigger step 2 of the aforementioned plan and attest the status of the FCU, even when no incorrect sensor data is detected.

In order to execute and validate the operational scenario described above, a stationary UAV-like setup has been designed, implementing the TEE-less environment with virtualization and trusted VMs instantiation of the CROSSCON Stack, as defined in section 3.1.4 of D2.1[3].

An overview of the prototype architecture, based on a real-world UAV implementation, is provided in the Figure 6:

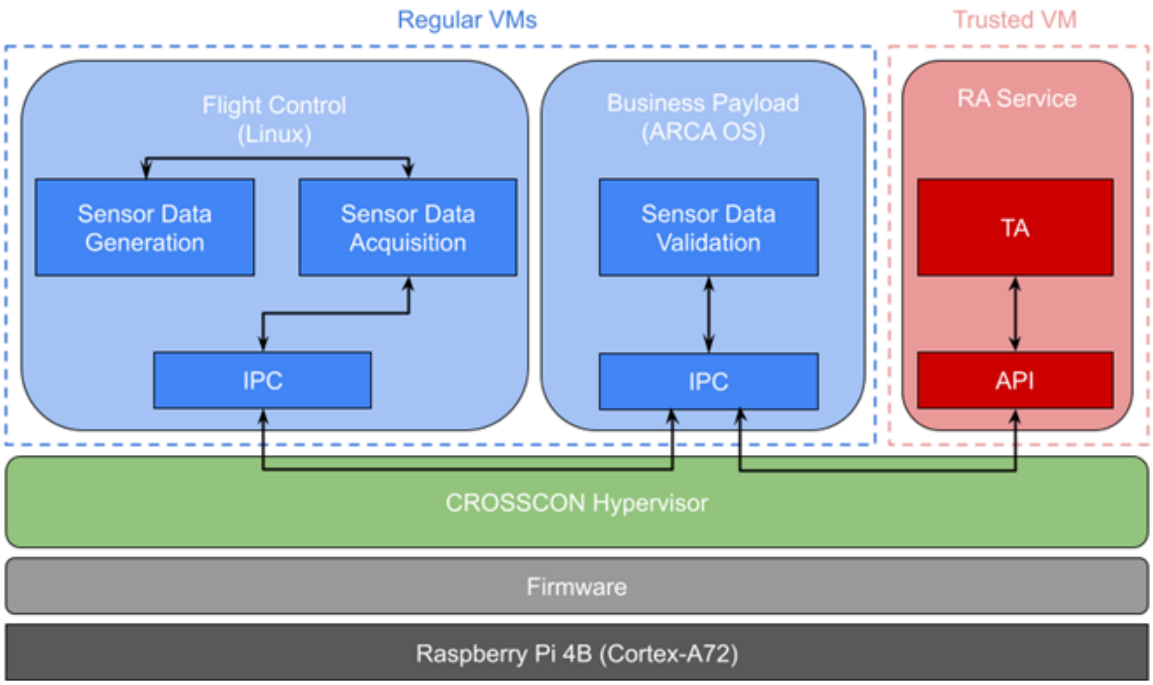

Figure 6: UC 4 prototype architecture & integration with CROSSCON Stack

More specifically, three main applications can be identified operating on top of the CROSSCON Hypervisor within the prototype architecture:

- **Flight Control:** Within the stationary UC4 prototype setup, the operational capabilities of the flight control have been omitted, focussing solely on sensor data handling:
    - A Sensor Data Generation module will be developed and deployed inside a dedicated Flight Control VM, which will emulate the output of a range of flight peripherals. Note that this emulated data generation module will be used to validate the initial implementation of the UC4 prototype, with the possibility to replace it with real-world sensors in a later stage of the project.
    - Alongside the aforementioned Sensor Data Generation module, a Sensor Data Acquisition module will be placed. Its operational functionality will be twofold: During nominal operations, it will simply retrieve the sensor data from the Sensor Data Generation module, and share it with the business payload VM, following the inter-process communication (IPC) implementation of CROSSCON, as defined in D2.1[3]. However, in the

scenario where the RA service needs to be triggered, it will also be able to adjust the sensor data before transmitting, in order to emulate a compromised FCU implementation.

- **Business Payload:** CYSEC's trusted operating system (OS), called ARCA OS, will be implemented to provide minimal protection and hardening of the Business Payload VM. Similar to the Flight Control VM, no operational business peripheral functionalities will be integrated into the Business Payload VM, focussing instead only on the verification of the sensor data obtained from the Flight Control VM. To achieve this, a Sensor Data Validation module will be implemented, responsible for (1) retrieving the sensor data from the Sensor Data Acquisition module via the IPC, (2) validation of the retrieved sensor data against a list of predefined critical values and (3) trigger the RA service through its API when the sensor data exceeds the aforementioned critical values.
- **RA Service:** Lastly, the RA service of the CROSSCON Stack will be implemented into the prototype, hosted as a Trusted Application within a Trusted VM, as described in D2.1[3]. It will be responsible for performing the attestation of the Flight Control VM and the secure communication with the RA verifier. As the development of the RA service is still ongoing, the exact API with which the RA service can be configured and operated will be defined at a later stage.
    - The RA verifier, required by the RA service described above, will be deployed alongside the prototype implementation depicted in Figure 6. However, as it will be operated independently and does not require integration with the CROSSCON Stack, it has been omitted from the detailed architecture.

3.4.2 Required Security Features

Before integration efforts for the UC 4 implementation prototype can be started, several components and trusted services introduced in Chapter 2 will need to be finalized:

- **CROSSCON Hypervisor support on RPi:** The implementation of UC4 depends on the ability to deploy and operate the CROSSCON Hypervisor on top of the Raspberry Pi 4B hardware platform. However, as described in Section 2.1, the CROSSCON Hypervisor is currently only compatible with QEMU, and efforts to ensure its compatibility with the required hardware platforms are ongoing.
- **RA Service:** The RA Trusted Application is central for the secure operation of the UC 4 prototype, ensuring the integrity of the Flight Controller. However, as described in Section 2.1, the RA service is currently still under development.
- **Secure Communication Channel:** To ensure the authenticity, integrity and confidentiality of the attestation report while it is being transmitted between the RA service and verifier, a method to establish a secure communication channel will have to be available within the RA service implementation.

3.4.3 Integration Timeline

Considering the proposed implementation architectures and operational scenarios, as well as the ongoing development of the CROSSCON Stack, the following integration timeline is predicted for the use case prototype:

Table 6: Integration timeline for UC 4

Integration Activity	2024						2025									
	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10
Integration of the CROSSCON Stack on Raspberry Pi 4B																
Development of the Flight Control & Business payload VMs																
Integration of the Remote Attestation TA																
Testing and validation activities																
Preparation of final demonstrator																

3.5 UC 5 - Intellectual Property Protection for Secure Multi-Tenancy on FPGA

3.5.1 Prototype Architecture

Field Programmable Gate Arrays (FPGAs) are versatile hardware platforms designed for customization. They comprise several key components, including configurable logic blocks (CLBs), block random access memory (RAM), and digital signal processing units (DSPs). These elements are interconnected through a programmable routing network (network of routing blocks (R) as shown in Figure 7), enabling the creation of any digital circuit configuration. The description of a digital circuit goes through several steps including, synthesis and place & route to produce the final binary file, referred to as the bitstream, that is to program the targeted FPGA.

FPGAs can be integrated with the processing system on a single chip, interconnected through on-chip buses, creating what is known as an FPGA-SoC (System on Chip). Alternatively, FPGAs can function as standalone peripherals, connected to the platform via external buses. This flexibility allows for diverse application scenarios, enhancing both performance and system design options.

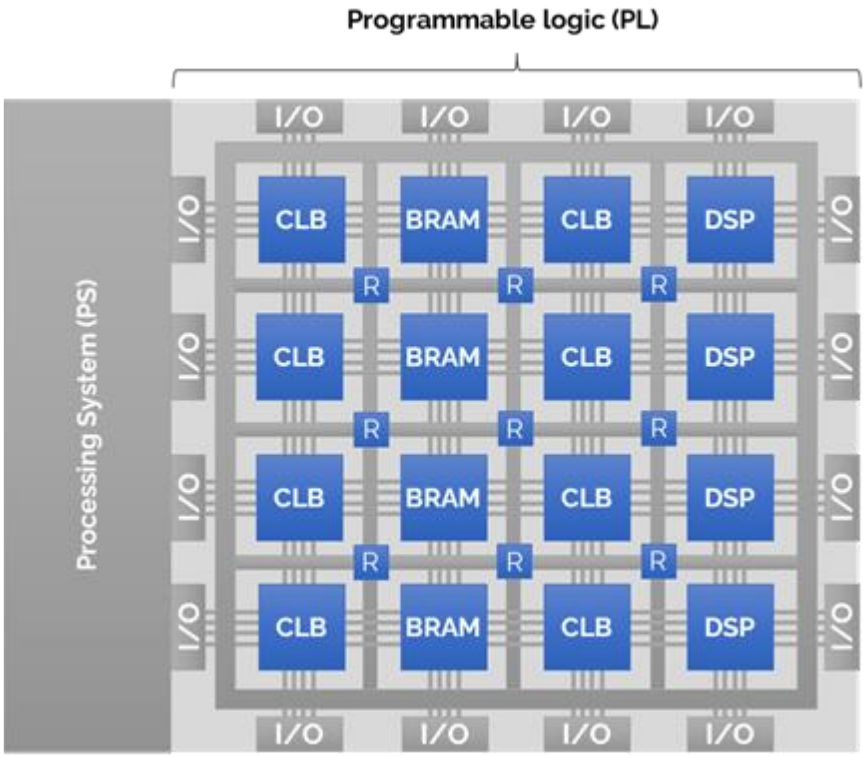

Figure 7: CROSSCON FPGA-SoC architecture & components

In UC5, our primary objective is to extend the trusted execution environment to FPGAs within a multi-tenant deployment model while maximizing FPGA utilization. To achieve this, we aim to enable secure sharing of FPGA resources among multiple users and applications, facilitating true multi-tenancy. This involves supporting secure configuration and deployment of intellectual property (IP) hardware designs on shared and virtualized FPGAs and enforcing access control to IP designs on the FPGA and their data. By doing so, we ensure that FPGA resources are utilized efficiently, providing robust security measures to protect each user's data and IP, and maintaining the integrity and confidentiality of all operations conducted on the FPGA platform.

In our prototype, we divide the physical FPGA into several virtual FPGAs, each equipped with its own set of resources (CLBs, BRAMs, DSPs, etc.). These virtual FPGAs can be managed independently, meaning they can be allocated, configured, and deallocated without affecting each other. This is made possible by leveraging the dynamic reconfigurability feature of FPGAs, which allows us to partition the FPGA into one static region and multiple reconfigurable regions (virtual FPGAs). This feature permits

the reconfiguration of any reconfigurable region at runtime, without impacting the other regions. Note that the static region can only be configured at boot time, i.e., after a power cycle.

In UC5, we partition the FPGA into two virtual FPGAs and a static region, as illustrated in Figure 8. The static region, referred to as the FPGA shell, abstracts the FPGA-specific details (such as I/O pins and interfaces) for users by providing standard I/O interfaces. Additionally, it handles the runtime configuration of the virtual FPGAs, ensuring efficient and secure management of the FPGA resources.

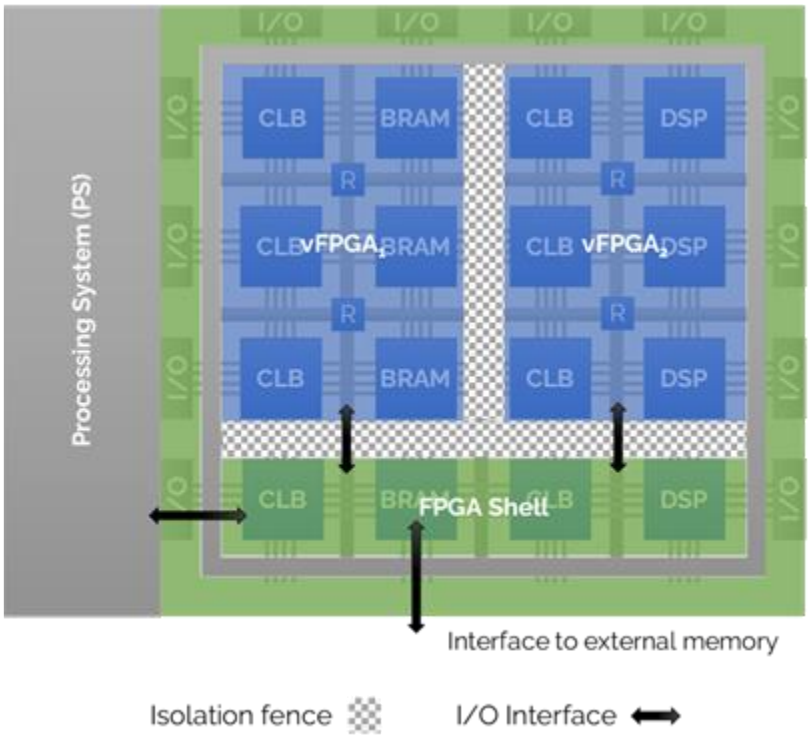

Figure 8: vFPGA partitions

To enforce isolation between the different regions on the FPGA, we utilize empty logic blocks to create an isolation fence, ensuring that signals from different regions are not routed adjacent to each other, thus preventing crosstalk-based attacks.

With the capability to create and manage different virtual FPGAs independently, we now focus on enabling the secure configuration of IP hardware designs, e.g., accelerators, on these virtual FPGAs. This involves implementing mechanisms for the decryption and verification of IP bitstreams before they are partially configured on the FPGA. Each user or IP provider should be able to encrypt and sign their IP bitstream with their own secret key. This secure configuration process involves encrypting and authenticating each IP hardware design using the application or user's secret key, safeguarding the integrity and confidentiality of the IP throughout its lifecycle.

Once the virtual FPGAs are successfully allocated and configured, strict access control measures must be enforced to ensure that only the user or application requesting the acceleration can communicate with and share data with the FPGA-based accelerator. This access control also extends to the FPGA shell, which should be accessed only by the trusted application that manages and provides FPGA services.

At system boot time, the FPGA shell and the isolation fence are configured on the FPGA, while the virtual FPGAs remain blank. This initial setup ensures a secure and isolated environment for subsequent operations.

In Figure 9, we illustrate our vision for integrating secure FPGA services into the CROSSCON Stack. These FPGA services are provided through a dedicated trusted application (TA_{FPGA}), ensuring robust security and management of the FPGA resources within the system. In the following we briefly discuss the functionality of TdA_{FPGA}.

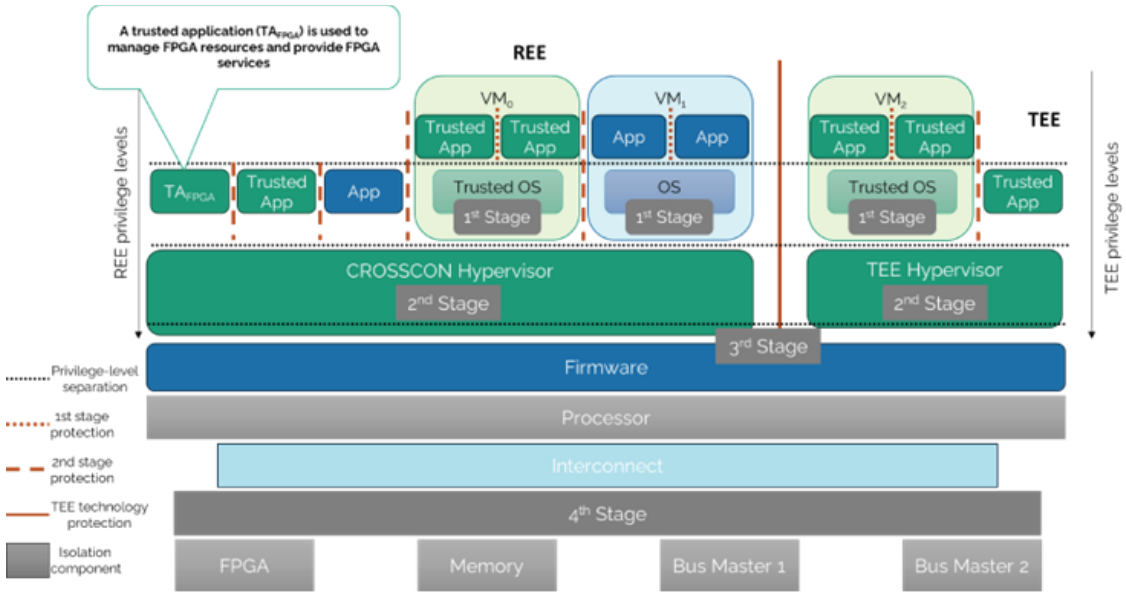

Figure 9: UC 5 integration with the CROSSCON Stack

TA_{FPGA} is responsible for the following activities:

1. **Managing Virtual FPGAs:** This involves the allocation and deallocation of virtual FPGAs to requesting applications. When an application (App x) sends a request to TA_{FPGA}, it checks for an available virtual FPGA (e.g., $vFPGA_1$). If available, TA_{FPGA} informs App x of the allocation. Deallocation can occur in two ways: either the virtual FPGA is allocated for a predefined time slot and then blanked using a special bitstream, or TA_{FPGA} notifies App x that the virtual FPGA will be revoked shortly.
2. **Session Key Exchange:** TA_{FPGA} and App x share a session key through a session key exchange using public key cryptography (e.g., RSA). This session key is used to encrypt the bitstream or to encrypt App x's secret key, which in turn encrypts and authenticates the bitstream. The encrypted bitstream is then shared with TA_{FPGA}. TA_{FPGA} provides a shared memory region for communicating bitstreams. This memory region allows other applications to write their encrypted bitstreams, authentication data, and tags, which TA_{FPGA} can read.
3. **Bitstream Decryption & Verification:** Once TA_{FPGA} receives the encrypted bitstream, it decrypts and verifies the bitstream before writing it to a private memory region shared with the FPGA shell. The FPGA shell then reads the bitstream and configures it on the intended virtual FPGA ($vFPGA_1$). For this step, we use AES-GCM mode with a 256-bit key size.
4. **Access Control Rules:** After successfully configuring the accelerator on the virtual FPGA, TA_{FPGA} provides access control enforcement to vFPGAs by only allowing requesting applications to communicate to their accelerators on the vFPGAs. Enforcing fine-grained access control to different FPGA resources, i.e. virtual FPGAs and the FPGA shell, can be also supported by the CROSSCON SoC through the Perimeter guard (PG). In our prototype, based on the AMD Xilinx ZCU102 board, we utilize Xilinx Peripheral Protection Units (XPPUs) and Xilinx Memory Protection Units (XMPUs) to complement the protection provided by MMUs and IOMMUs.

Currently, steps 1 and 3 have been implemented, while steps 2 and 4, along with integration within the CROSSCON Stack, are work in progress.

3.5.2 Required Security Features

The trusted FPGA services rely on the CROSSCON Stack and the underlying TEE technology to provide isolation for the TA_{FPGA}. More specifically, the implementation of UC 5 requires:

- **CROSSCON Hypervisor support:**
    - The implementation of UC 5 depends on the ability to deploy and operate the CROSSCON Hypervisor on top of the selected AMD Xilinx ZCU102 board. However, as described in Section 2.1, the CROSSCON Hypervisor is currently only compatible with QEMU, and

efforts to ensure its compatibility with the different required hardware platforms are ongoing.

- Service discovery mechanisms should be put in place for applications to locate TA_{FPGA} and communicate with it using its identifier.
- The CROSSCON Hypervisor should allow TA_{FPGA} to configure isolation components, e.g., MMUs and IOMMUs, to control access to virtual FPGAs and their associated memory regions.

- **API implementation:** The design of a clear and efficient API is crucial for facilitating communication between applications and TA_{FPGA}, including functions for requesting FPGA resources, managing job submissions, and retrieving results.

3.5.3 Integration Timeline

Considering the proposed implementation architectures and operational scenarios, as well as the ongoing development of the CROSSCON Stack, the following integration timeline is predicted for the use case prototype:

Table 7: Integration timeline for UC 5

Integration Activity	2024						2025									
	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10
Finalization of missing dependencies																
Finalization of APIs																
Integration with the CROSSCON Stack																
Testing and validation activities																
Preparation of final demonstrator																

4 Conclusions

This document has provided an overview regarding the initial integration efforts of the CROSSCON Stack, as preparation of the testing and validation activities of T5.3 and T5.4, respectively.

Chapter 2 detailed the development status of each of the CROSSCON Stack's components and trusted services. Although development of some of these core parts of the CROSSCON Stack is still ongoing, most of the services required for the integration of the first use case prototypes were demonstrated to be either ready for integration, or in the final stages of development. As such, we are confident that the integration activities will advance as planned.

As part of the next steps of the integration activities, Chapter 3 provided a detailed overview of each of the use case prototypes, including their implementation architectures, operational workflows, required security features and predicted integration timeline.

References

[1] **H. Koshutanski**, **D1.4** Use Cases Definition Final Version, 2023.

[2] **A. García**, **D1.5** Requirements Elicitation Final Technical Specification, 2024

[3] **M. Miettinen, S. Zeitouni**, **D2.1** CROSSCON Open Specification – Draft, 2023.

[4] **S. Pinto**, **D3.1** CROSSCON Open Security Stack Documentation - Draft, 2024.

[5] **S. Pinto**, **D3.2** CROSSCON Open Security Stack – Initial Version, 2024.

[6] **Ž. Putrle, S. Zeitouni**, **D4.1** CROSSCON Extensions to Domain Specific Hardware Architectures Documentation – Draft, 2024.

[7] **Ž. Putrle**, **D4.2** CROSSCON Extension Primitives to Domain Specific Hardware Architectures — Initial Version, 2024.

[8] https://github.com/bao-project/bao-hypervisor

[9] https://www.qemu.org/

[10] **TEE client API**
https://optee.readthedocs.io/en/latest/architecture/globalplatform_api.html#tee-client-api

[11] **TEE internal Core API**
https://optee.readthedocs.io/en/latest/architecture/globalplatform_api.html#tee-internal-core-api

[12] **RFC 9019** https://datatracker.ietf.org/doc/html/rfc9019