GLOBALPLATFORM®
THE STANDARD FOR SECURE DIGITAL SERVICES AND DEVICES

GlobalPlatform Technology
MCU Root of Trust Protection Profile
Version 1.0

Public Release

December 2022

Document Reference: GPT_SPE_146

THIS SPECIFICATION OR OTHER WORK PRODUCT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE COMPANY, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT.

Contents

Figures

Tables

1 Introduction

Title:	MCU Root of Trust Protection Profile
Identification:	GPT_SPE_146 (Core MCU RoT PP)
Date:	December 2022
Version:	1.0
Sponsor:	GlobalPlatform, Inc.
CC Version:	3.1 Revision 5

Title:	MCU RoT Persistent Time PP-Module
Identification:	GPT_SPE_160
Date:	December 2022
Version:	1.0
Sponsor:	GlobalPlatform, Inc.
CC Version:	3.1 Revision 5

Title:	MCU RoT Debug PP-Module
Identification:	GPT_SPE_161
Date:	December 2022
Version:	1.0
Sponsor:	GlobalPlatform, Inc.
CC Version:	3.1 Revision 5

Title:	MCU RoT ARoT Isolation PP-Module
Identification:	GPT_SPE_162
Date:	December 2022
Version:	1.0
Sponsor:	GlobalPlatform, Inc.
CC Version:	3.1 Revision 5

The aim of this document is to define the security requirements for an architecture neutral microcontroller Root of Trust (MCU RoT) which enables IoT-oriented security services such as attestation, device authentication, personal data protection, etc. It was developed based on GlobalPlatform Trusted Execution Environment Protcetion Profile (TEE PP) by the PSA JSA group then donated to the GlobalPlatform TEE Committee, which has finalized the document. This Protection Profile (PP) constitutes the reference for the evaluation of the MCU RoT both in the Common Criteria (CC) and the GlobalPlatform schemes.

This document defines:

- The core MCU RoT Protection Profile (PP), which includes the minimum MCU RoT security requirements, such as secure initialisation, firmware integrity, secure storage, isolation of the Secure Partition Manager (SPM), isolation of the MCU RoT services, etc.
- Three complementary PP-Modules:
    - MCU RoT Persistent Time PP-Module, or simply Persistent Time PP-Module, for persistent monotonic time
    - MCU RoT Debug PP-Module, or simply Debug PP-Module, for controlled access to debug features
    - MCU RoT ARoT Isolation PP-Module, or simply ARoT Isolation PP-Module, for isolation between Application Roots of Trust.

The core MCU RoT PP can be used alone or in a PP-Configuration with any subset of one, two, or three PP-Modules. This document implicitly defines such PP-Configurations.

For the sake of simplicity, this document is referred to as a PP.

This PP claims conformance with the assurance package EAL 2+ which consists of the predefined EAL 2 augmented with ALC_FLR.2 and the extended assurance component AVA_VAN_AP.3 that requires resistance to Enhanced-basic attack potential. The Common Evaluation Methodology defined in [CEM] applies to all the standard assurance components. The attack potential table to use for AVA_VAN_AP.3 is defined in [TEE PP].

1.1 Audience

This document is intended primarily for the use of MCU developers and integrators, service providers (ARoT developers), as well as evaluation laboratories, certification bodies, and consumers of GlobalPlatform and Common Criteria certificates.

1.2 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of intellectual property rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit https://globalplatform.org/specifications/ip-disclaimers/. GlobalPlatform shall not be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence of the evidence, validity, or scope of any such IPR.

1.3 References

Table 1 and Table 2 present the references that are applicable to this Protection Profile. The latest version of each reference applies unless a publication date or version is explicitly stated.

Table 1: Normative References

Standard / Specification	Description	Ref.
CC Part 1	Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1, revision 5, April 2017. CCMB-2017-04-001.	[CC1]
CC Part 2	Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1, revision 5, April 2017. CCMB-2017-04-002.	[CC2]
CC Part 3	Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements. Version 3.1, revision 5, April 2017. CCMB-2017-04-003.	[CC3]
CEM	Common Methodology for Information Technology Security Evaluation, Evaluation Methodology. Version 3.1, revision 5, April 2017. CCMB-2017-04-004.	[CEM]
GPD_SPE_021	GlobalPlatform Technology TEE Protection Profile v1.3 (or Latest applicable version)	[TEE PP]
GPD_TEN_081	GlobalPlatform Technology Cryptography Recommendations for TEE Internal Mechanisms	[CRec]
BSI AIS 31	A proposal for: Functionality classes for random number generators. Version 2.0, 18 September 2011	[AIS31]
FIPS Publication	ADVANCED ENCRYPTION STANDARD (AES). FIPS PUB 197. November 2011.	[AES]
FIPS Publication	DATA ENCRYPTION STANDARD (DES). FIPS PUB 46-3. October 1999.	[DES]
RSA Laboratories Publication	RSA Cryptographic Standard. PCKS#1 v2.2. October 2012	[RSA]
FIPS Publication	SECURE HASH STANDARD. FIPS PUB 180-4. March 2012	[SHA]
IEEE Standard	IEEE 1149.1-2001 Standard Test Access Port and Boundary-Scan Architecture http://standards.ieee.org/reading/ieee/std_public/description/testtech/1149.1-2001_desc.html	[JTAG]
NIST Special Publication 800-90B	Recommendation for the Entropy Sources Used for Random Bit Generation. January 2018	[SP800-90]
RFC 2119	Key words for use in RFCs to Indicate Requirement Levels	[RFC 2119]
RFC 4122	A Universally Unique IDentifier (UUID) URN Namespace	[RFC 4122]

Table 2 Informative References

Standard / Specification	Description	Ref.
ARM DEN_0079	PSA Security Model	[ARM DEN 0079]

1.4 Terminology and Definitions

The following meanings apply to SHALL, SHALL NOT, MUST, MUST NOT, SHOULD, SHOULD NOT, and MAY in this document (refer to [RFC 2119]):

- **SHALL** indicates an absolute requirement, as does **MUST**.
- **SHALL NOT** indicates an absolute prohibition, as does **MUST NOT**.
- **SHOULD** and **SHOULD NOT** indicate recommendations.
- **MAY** indicates an option.

Selected terms used in this document are included in Table 3. Additional terms are defined in Common Criteria [CC1].

Table 3: Terminology and Definitions

Term	Definition
Application Programming Interface (API)	A set of rules that software programs can follow to communicate with each other.
Application Root of Trust (ARoT)	An application running inside the Secure Processing Environment that exports security related functionality to Client Applications outside of the MCU RoT. Contrast *Client Application*.
ARoT instance time / ARoT persistent time	Time value available to an Application Root of Trust through the API. The API offers two types of time values: • System Time exists only during runtime and must be monotonic for a given ARoT instance. The returned value is called “ARoT instance time”. • Persistent Time persists over resets, depends only on the ARoT but not on a particular instance, and must be monotonic even across power cycles. Its monotonicity across power cycles is related to the optional MCU RoT Persistent Time PP-Module.
Client Application (CA)	An application running in the Non-Secure Processing Environment (NSPE) that accesses facilities provided by Application Roots of Trust or the Root of Trust Services inside the SPE. Contrast *Application Root of Trust*.
Device binding	The property of data being usable only on a unique given system instance, here an MCU RoT.
Execution Environment (EE)	An environment that hosts and executes software. This could be a REE, with hardware hosting Android, Linux, Windows, an RTOS, or other software; it could be a Secure Element or a TEE.

Term	Definition
Hardware External Interface	In the context of this document, the package input and output interfaces, which provide access to the package resources and indirectly to the SoC internals.
Hardware Unique Key (HUK)	A persistent key, provisioned during device manufacture, used to bind client and device specific secrets to the MCU RoT.
Integrity	Property of the persistent storage that means that the value successfully read from a storage location is the last value that was written to this location.
Monotonicity	The property of a variable whose value is either always increasing or always decreasing over time.
Non-Secure Processing Environment (NSPE)	An Execution Environment comprising at least one NSPE OS and all other components of the device (SoCs, other discrete components, firmware, and software) which execute, host, and support the NSPE OS (excluding any Secure Components included in the device). From the viewpoint of a Secure Component, everything in the NSPE is considered untrusted, though from the NSPE OS point of view there may be internal trust structures. Contrast *Secure Processing Environment (SPE)*.
NSPE Client API	The software interface used by clients running in the NSPE to communicate with the MCU RoT and with the Application Roots of Trust executed on the SPE.
NSPE Communication Agent	An NSPE OS driver that enables communication between the NSPE and the SPE. Contrast *SPE Communication Agent*.
NSPE OS	An OS executing in an NSPE. May be anything from a large OS such as Linux down to a minimal set of statically linked libraries providing services such as a TCP/IP stack. Contrast *Secure Partition Manager*.
Power cycle	The lapse between the moment a device is turned on and the moment the device is turned off afterwards.
Root of Trust (RoT) / MCU RoT	A computing engine, code, and possibly data, all co-located on the same platform; provides security services. In the framework of this document, MCU RoT designates a RoT implemented in an MCU.
RoT Internal API / MCU RoT Internal API	The software interface exposing the RoT functionality to Application Roots of Trust. In the framework of this document, MCU RoT Internal API and Internal API are used.
SPE Communication Agent	An SPM driver that enables communication between the NSPE and the SPE. Contrast *NSPE Communication Agent*.

Term	Definition
Secure Partition Manager (SPM)	The OS running in the SPE. It manages the isolation of MCU RoT services, the IPC mechanism that allow software in one domain to make requests of another, and scheduling logic to ensure that requests are serviced. Contrast *NSPE OS*.
Secure Processing Environment (SPE)	An Execution Environment that runs alongside but isolated from an NSPE. An SPE has security capabilities and meets certain security-related requirements: it protects its assets against a set of defined threats which include general software attacks as well as some hardware attacks, and defines rigid safeguards as to data and functions that a program can access. There are multiple technologies that can be used to implement an SPE. Contrast *Non-Secure Processing Environment.*
Software External Interface	In the context of this document, the MCU RoT Internal API (used by the ARoTs) and the Client API (used by the NSPE).
System-on-Chip (SoC)	An electronic system all of whose components are included in a single integrated circuit.
Trusted Storage	Storage that is protected either by the hardware of the SPE, or cryptographically by keys held in the SPE. If keys are used they are at least of the strength used to instantiate the SPE. An SPE Trusted Storage is not considered hardware tamper resistant to the levels achieved by Secure Elements.

1.5 Abbreviations and Notations

Table 4 defines abbreviations used within this Protection Profile.

Table 4: Abbreviations and Notations

Abbreviation	Meaning
AES	Advanced Encryption Standard (defined in [AES])
API	Application Programming Interface
ARoT	Application Root of Trust
CA	Client Application
CC	Common Criteria (defined in [CC1], [CC2], [CC3])
CEM	Common Evaluation Methodology (defined in [CEM])
EAL	Evaluation Assurance Level (defined in [CC1])
EE	Execution Environment
HUK	Hardware Unique Key
ID	IDentifier
JTAG	Joint Test Action Group (defined in [JTAG])
MCU	Microcontroller Unit

Abbreviation	Meaning
NA	Not Applicable
NSPE	Non-Secure Processing Environment
OS	Operating System
OSP	Organisational Security Policy (defined in [CC1])
OTP	One-Time Programmable
PCB	Printed Circuit Board
PP	Protection Profile (defined in [CC1])
RAM	Random Access Memory
RFC	Request For Comments; may denote a memorandum published by the IETF
ROM	Read Only Memory
RoT	Root of Trust
RSA	Rivest / Shamir / Adleman asymmetric algorithm (defined in [RSA])
RTOS	Real Time Operating System
SAR	Security Assurance Requirement (defined in [CC1])
SFP	Security Function Policy (defined in [CC1])
SFR	Security Functional Requirement (defined in [CC1])
SHA	Secure Hash Algorithm (defined in [SHA])
SoC	System-on-Chip
SPD	Security Problem Definition (defined in [CC1])
SPE	Secure Processing Environment
SPM	Secure Partition Manager
ST	Security Target (defined in [CC1])
TDES	Triple DES (Data Encryption Standard)
TOE	Target of Evaluation (defined in [CC1])
TSF	TOE Security Functionality (defined in [CC1])
USB	Universal Serial Bus

1.6 Revision History

Table 5 presents the major releases of the present document.

Table 5: Revision History

Date	Version	Description
December 2nd, 2022	1.0	Public Release

2 TOE Overview

This chapter defines the type of the Target of Evaluation (TOE), presents typical TOE architectures, and describes the TOE's main security features and intended uses as well as the TOE's life cycle.

2.1 TOE Type

The TOE type is the Root of Trust (RoT) for microcontrollers (MCUs) or simply an MCU RoT. It is based on the security principles described in the PSA Device Security Model [ARM DEN 0079]. However, this PP does not mandate functional compliance with any specification.

The TOE is an execution environment isolated from any other execution environment, including the usual Non-Secure Processing Environment (NSPE), and their applications. The TOE may host a set of Application Roots of Trust (ARoTs) and provides them with a comprehensive set of security services including integrity of execution, trusted storage, key management and cryptographic algorithms, time management, and attestation.

The TOE, as depicted in Figure 2-1, comprises:

- Any hardware, firmware, and software used to provide the MCU RoT security functionality. This includes:
  - Immutable MCU Root of Trust, for example Boot ROM, parameters such as initialisation data, hardware keys, and IDs, isolation hardware, security life cycle management and enforcement (such as the Debug feature if present). This component cannot be updated.
  - Updateable MCU Root of Trust, such as software isolation framework protecting more trusted software from less trusted software, generic services such as binding, initial attestation, generic crypto services, firmware update validation.
  - Trusted Subsystems used by the MCU Root of Trust, such as security subsystems, trusted peripherals, SIM or SE, including both hardware and software components.
- The guidance for the secure usage of the MCU RoT after delivery.

The TOE does not comprise:

- The Application Roots of Trust
- The Non-Secure Processing Environment
- The Applications hosted by the NSPE.

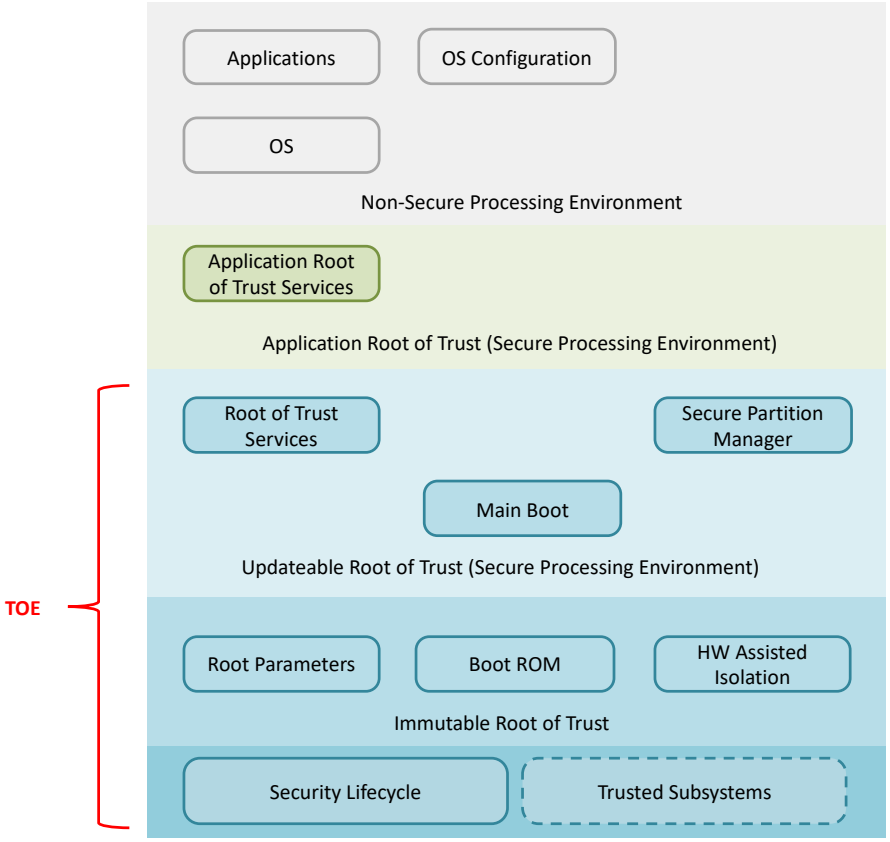
If the MCU RoT implements debug interfaces, then either these must be disabled in end-usage phase (also called production mode) or the debug feature must comply with the MCU RoT Debug PP-Module defined in this document.

The form-factor of the TOE is either a SoC or a Final Device.

Application Note:

The Security Target (ST) author shall determine and describe the form-factor that corresponds to the TOE.

Figure 2-1: Scope of Evaluation



In the remainder of this document, TOE and MCU RoT are used interchangeably.

2.2 TOE Description

2.2.1 Software Architecture of a Device enabled with an MCU RoT

The MCU RoT (the TOE) is embedded in the device and runs alongside a standard Non-Secure Processing Environment, such as a Real Time OS (RTOS). Figure 2-2 provides a high-level view of the software components of an MCU RoT device and the TOE, independently of any hardware architecture.

The TOE software architecture identifies three distinct classes of components:

- The Root of Trust Services that provide security services to the SPE and NSPE
- The Internal APIs
- The Application Roots of Trust that run on the SPE and access its services through the Internal APIs.

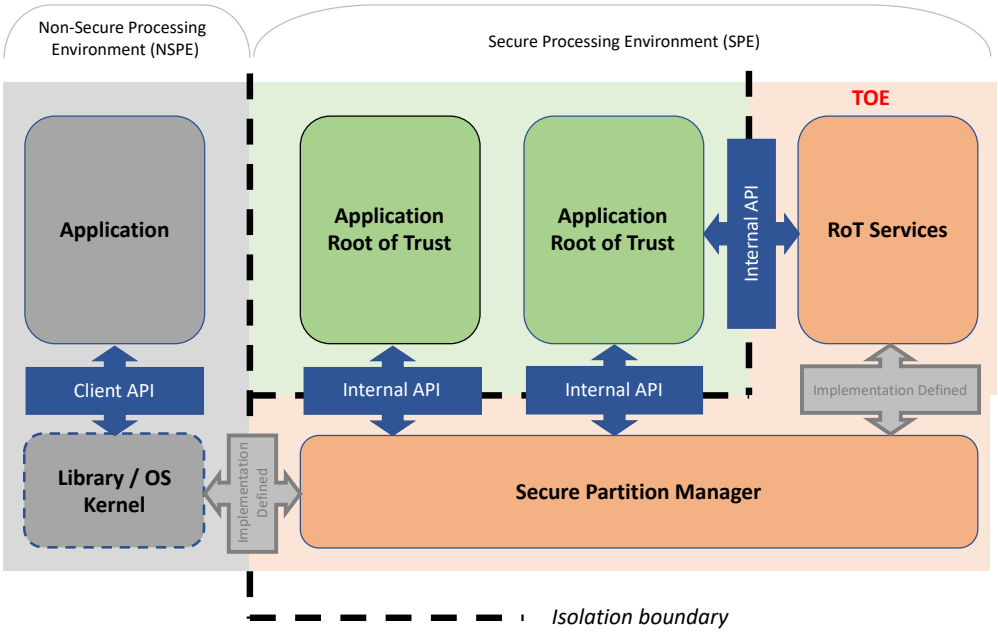
The NSPE software architecture also identifies three distinct classes of components:

- The NSPE OS, which provides the Client API and sends requests to the SPE
- The Client API
- The NSPE Applications which use the Client API to access the secure services offered by the MCU RoT running on the SPE.

The TOE Software External Interface comprises the Internal API (used by the Application RoTs) and the SPE Communication Agent.

The communication protocol between the NSPE and the SPE, used below the Client API level, is implementation-dependent, and therefore this PP does not mandate any protocol.

Figure 2-2: MCU RoT Software Architecture



Application Note:

The ST author shall identify and describe all the software interfaces used for communication from and to the SPE.

2.2.2 Hardware Architecture of Device enabled with an MCU RoT

The TOE is embedded in a device platform including:

- Microcontroller processing unit(s)
- Hardware resources such as:
  - Physical volatile memory
  - Physical non-volatile memory
  - Peripherals, such as network devices
  - Cryptographic accelerators
  - Secure hardware clock
- A set of connections between the processing unit(s) and the hardware resources.

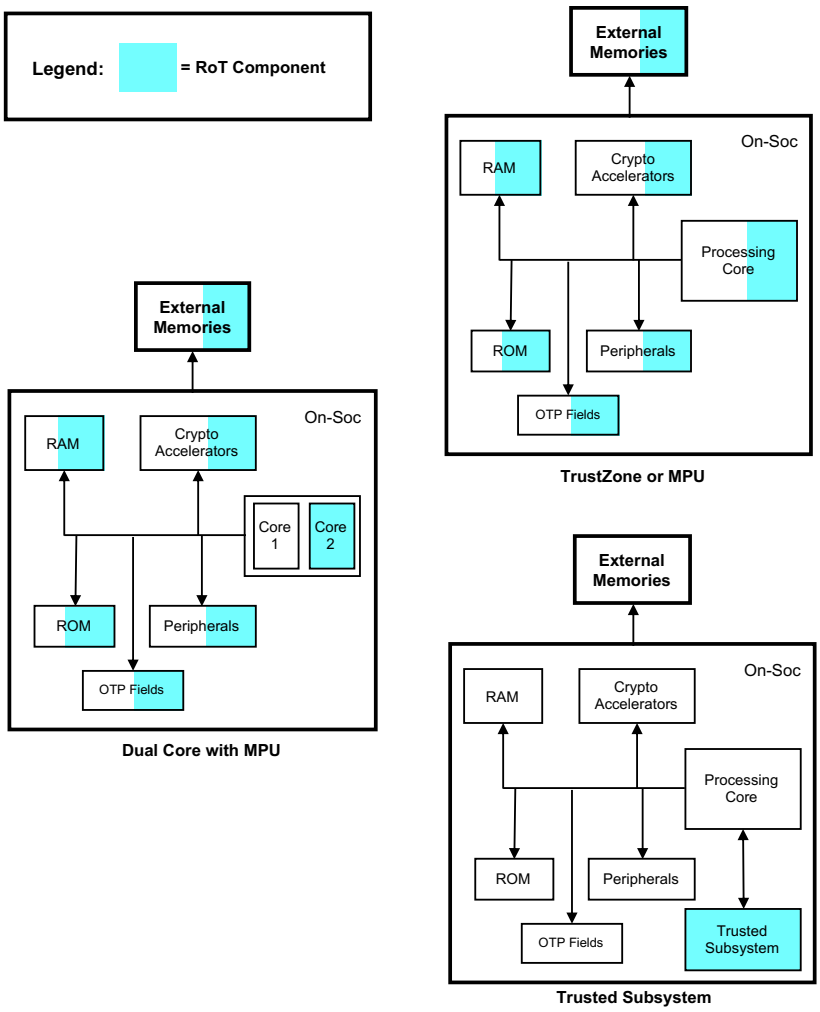
Schematically, a device enabled with an MCU RoT is structured in three layers:

- The *die layer*, System-on-Chip (SoC), which contains processor(s) and resources such as memories, crypto-accelerators, peripherals (e.g. JTAG, USB, serial), etc.

- The *package layer*, which embeds the SoC and contains further resources, e.g. non-volatile and volatile memories, pins, or buses. Resources inside the same package layer are connected using buses that are not externally accessible. External buses in the specification are outside the package layer. 3D die stacking techniques may be used to place more facilities inside the package that may not be in the die layer.
- The *PCB layer*, which contains the package layer, additional non-volatile and volatile memories, wireless and contactless interface chips, and other resources.

The TOE is typically implemented in the die and package layers of one package, but it may be instantiated in several separate packages using cryptographic linking (secure channels) between components.

Figure 2-3: Examples of MCU RoT Realizations



The Hardware External Interface of the TOE comprises the package input and output interfaces, which provide access to the package resources and indirectly to the SoC internals. This PP considers the package internals as opaque.

Nevertheless, the physical boundary of the TOE depends on its implementation. Furthermore, the set of "trusted" hardware resources used to realize the security functionality, which is controlled by the MCU RoT, can change dynamically during the execution of the MCU RoT Services. For instance, some communication resources such as the network device may sometimes be within the TOE boundaries if the TOE enforces exclusive access to these resources. From a logical point of view, the "trusted" resources used by the TOE

are separated from the “untrusted” resources used by the NSPE. That is, the MCU RoT (in SPE) and the NSPE coexist in the device but are hardware-isolated from each other.

In practice, there are several possible architectures of an MCU RoT running isolated from the NSPE. Figure 2-3 illustrates three possible realizations, with different resource-sharing policies between the MCU RoT in the SPE and the NSPE. Indeed, the SPE and the NSPE can share device resources provided that the MCU RoT controls access to them.

This PP does not mandate any hardware architecture, resource set, or isolation mechanisms from the NSPE. STs conformant to this PP shall describe the physical layout and precisely define the physical boundaries of the MCU RoT and its Hardware External Interface.

2.3 Usage and Major Security Features of the TOE

The purpose of the TOE is to provide security services and to host and execute Application RoTs securely, enforcing their mutual isolation and isolation from other execution environments, and ensuring integrity and confidentiality of the assets managed by the TOE.

The following sections define the TOE security functionality and the TOE intended usage.

2.3.1 TOE Security Functionality

The TOE security functionality in the end-usage phase (cf. section 2.5) which is in the scope of evaluation consists of:

- Secure MCU RoT instantiation, through a secure initialisation process using assets bound to the SoC, that ensures the authenticity and contributes to the integrity of the MCU RoT code running on the device
- Isolation of the Secure Partition Manager (SPM), MCU RoT services, the MCU RoT resources involved, and each Application Root of Trust from the NSPE
- Isolation of the SPM and MCU RoT services from Application Roots of Trust
- Trusted storage of ARoT and MCU RoT data and keys, ensuring integrity, confidentiality, atomicity, and binding to the MCU RoT
- Random Number Generation
- Cryptographic Operations API, e.g. generation and derivation of keys and key pairs, support for cryptographic algorithms such as SHA-256, AES 128/256, TDES, RSA 2048, etc.
- ARoT instantiation that ensures the authenticity and contributes to the integrity of the ARoT code
- Initial attestation service, to declare to a third-party verification service how the combination of hardware and firmware of the TOE can be trusted
- Monotonic ARoT instance time
- Correct execution of SPM and MCU RoT services
- MCU RoT firmware integrity verification
- Prevention of downgrade of MCU RoT firmware.

The MCU RoT security functionality defines the logical boundary of the TOE. The interfaces of this boundary consist of the Software External Interface and the Hardware External Interface, introduced in sections 2.2.1 and 2.2.2 respectively. The security functionality provided by the Application RoTs is out of scope of the TOE.

Application Note:

The ST author shall complete the descriptions of the security functionality with the characteristics of the actual TOE, including any ARoT management functionalities if applicable (on top of verification of ARoT authenticity prior to execution), and the complete list of cryptographic algorithms supported by the product.

2.3.2 TOE Usage

The MCU RoT enables the use of IoT devices for a wide range of services that require security protection, for instance:

- Metering: Such devices require protection of the integrity (and confidentiality) of data.
- User authentication: Such services require a robust Root of Trust, isolation from other execution environments, and controlled use of the user interfaces, such as biometric sensors and displays.
- Data protection: Devices store increasing amounts of personal information (such as contacts, messages, photos, and video clips) and sensitive data (credentials, passwords, health data, etc.). Secure storage means are required to prevent exposure of this information in the event of loss, theft, or any other adverse event, such as malware.
- Infrastructure for IoT Edge.

2.3.3 MCU RoT Persistent Time PP-Module

The MCU RoT Persistent Time PP-Module addresses the following additional security functionality:

- Monotonic ARoT persistent time.

This PP-Module applies when the TOE can maintain a monotonic persistent time over reset and allows ARoT applications to obtain such time values.

Note that monotonic persistent time allows a service depending on a notion of time to be delivered after a power cycle without any remote help to synchronize time. For connected services that can get an updated time at startup, monotonic instance time may be sufficient.

2.3.4 MCU RoT Debug PP-Module

The MCU RoT Debug PP-Module addresses the following additional security functionality:

- MCU RoT Debug interface.

This PP-Module applies to deployed TOEs that provide debug means in the end-usage phase, in which case they shall be accessible to authorized users only.

2.3.5 MCU RoT ARoT Isolation PP-Module

The MCU RoT ARoT Isolation PP-Module addresses the following additional security functionality:

- Isolation between Application Roots of Trust.

This PP-Module applies to TOEs that provide an isolated runtime environment to the applications running in the MCU.

2.4 Available Non-TOE Hardware/Software/Firmware

The TOE may require some non-TOE hardware, software, or firmware in order to operate, such as non-volatile memory. However, the TOE must be realized in a way such that TOE security functionalities do not rely on proper behaviour of non-TOE hardware, software, or firmware.

Application Note:

The ST author shall provide a complete description of the available non-TOE hardware/software/firmware with the list of non-TOE resources used by the TOE.

2.5 Reference TOE Life Cycle

The device life cycle outlined here is a reference life cycle which is split into seven phases:[1]

- Phase 1 corresponds to the design of firmware, software, and hardware; it covers the MCU RoT, and additional components.
- Phase 2 corresponds to the overall design of the hardware platform supporting the MCU RoT.
- Phase 3 corresponds to chipset and other hardware components manufacturing.
- Phase 4 covers software preparation (e.g. linking the MCU RoT software and other software).
- Phase 5 consists of TOE assembly; it includes any initialisation and configuration step necessary to bring the TOE to a secure state prior to delivery to the end-user.
- Phase 6 stands for the end-usage of the TOE.
- Phase 7 covers the case where the TOE is returned to the manufacturer for failure analysis.

Secure boot/firmware, including MCU RoT initialisation code, is usually installed in phase 3 though it may be upgraded later. The Hardware Unique Key (HUK) and the MCU RoT unique identifier (see definition of assets in section 4.1.1) are set by injection or on-board creation in phase 3 or 5. The SPM and MCU RoT Services are usually installed after this step, in phase 3 or 5, though they may be upgraded later. Application Roots of Trust may be installed with or after the SPM, either in phase 3 or in phase 5 – Application RoTs installed in phase 5 may have been linked with the SPM in phase 4. If the TOE supports Debug functionalities, the flag to indicate whether the functionality is enabled and the Debug credentials (such as a Debug authentication key) are set in phase 3 or 5.

The TOE delivery point establishes the limits of the evaluation: the delivery point can range from the end of phase 3 to the end of phase 5, but must necessarily follow the setting of the HUK and the MCU RoT unique identifier; the Debug enabled flag and the Debug credentials (if the MCU RoT Debug PP-Module is included); and the SPM installation:

- The security of the environments, processes, and procedures before the delivery point is evaluated according to the EAL 2 through the ALC assurance class.
- The security of the environments, processes, and procedures from the delivery point up to end of phase 5 is covered through the AGD assurance class by organisational security policies and security objectives for the environment.

[1] Actual development, manufacturing, and assembly processes can deviate from this reference life cycle.

- The security of the end-usage environment is covered by the TOE security functionalities and by security objectives for the environment.

Table 6 presents a possible instantiation of the actors involved in the different life cycle phases. Note that actors may delegate operations to other entities provided the overall security level is met.

For the sake of readability, the table is not meant to cover all possibilities. Other flows, consisting of the seven same phases, but with different ownership of the phases represented, are possible. Cases where the SPE runs on a discrete separate processor, or where the MCU RoT is installed by the chipset manufacturer and the device manufacturer merely integrates a turn-key platform that the chipset manufacturer provides, or on the contrary where the device manufacturer is fully responsible for MCU RoT integration, can result in different flows.

Table 6: Actors in the TOE Life Cycle

Phases	Actors
1 & 2: Firmware / Software / Hardware design	The MCU RoT software developer: • Is in charge of the software development and testing; • May also develop the initialisation code that instantiates/initialises the MCU RoT (e.g. part of the secure boot code); • Specifies the MCU RoT software linking requirements. The device manufacturer: • May design NSPE software to be linked with the MCU RoT in phase 4 to provide NSPE-controlled resources; • May design Application Roots of Trust and integrate these in phase 4. The MCU RoT hardware designer designs (part of) the processor(s) where the software runs and (part of) the hardware security resources. The chip vendor: • Designs the immutable boot code (ROM code) and the secure portion of the chipset; • If not designing the full MCU RoT hardware, integrates (and potentially augments) the MCU RoT hardware designed by third-party hardware designer(s).
3: Chip manufacturing	The chip vendor produces the chipset including the TOE. At this point, the TOE must be fully testable to permit checking for manufacturing defects. The TOE is then configured in multiple phases by the chip vendor and the device manufacturer (phase 4), e.g. by the programming of fuses to enable, set or seed the Hardware Unique Key.
4: Software manufacturing	The device manufacturer is responsible for the integration, validation, and preparation of the software to load in the product, including the MCU RoT Services, possibly some pre-installed Application Root(s) of Trust, and additional software required to use the product (e.g. NSPE, Client Applications).
5: Device manufacturing	The device manufacturer is responsible for the device assembly, initialisation, provisioning, and any other operation on the TOE (including loading or installation of Application Roots of Trust) and the device before delivery to the end user. The debug features of the device are either disabled or they are part of the TOE.

Phases	Actors
6: Deployed (end-usage phase)	The end user receives a ready-to-use device that includes the TOE.
7: Return Material Authorisation (RMA)	This is a terminal state used for devices that are returned to the device manufacturer for failure analysis. When a device is put into the RMA state, it loses access to its secret keys and, with it, the ability to operate securely.

Application Note:

- The ST author shall describe the actual TOE life cycle, identify the actors and development/manufacturing sites involved (including identifying the actual integration points of the components (SPM and MCU RoT services, HUK, ARoTs) into the TOE, as well as the actual delivery point of the TOE, and specify the process for setting the Hardware Unique Key and the phase in which it occurs.
- The ST author shall also identify the TOE and the components that are delivered with the TOE if any, e.g. pre-installed Application Roots of Trust. If the TOE provides ARoT management functionality (i.e. installation of ARoTs in phase 6 or in general after the delivery point), which is not in the scope of this PP, it must be described in the ST as well. Note that this PP does not mandate any particular ARoT life cycle management policy.

3 Conformance Claims and Consistency Rationale

3.1 Conformance Claim to CC

This Protection Profile claims conformance to CC Part 2 [CC2] and CC Part 3 [CC3] extended.

CC Part 2 is extended with the security functional components ‘FCS_RNG.1 Random numbers generation’ and ‘FPT_INI.1 TSF initialisation’.

CC Part 3 is extended with the security assurance component ‘AVA_VAN_AP.3 Vulnerability analysis’.

The MCU RoT Persistent Time, MCU RoT Debug, and MCU RoT ARoT Isolation PP-Modules are CC Part 2 [CC2] conformant.

3.2 Conformance Claim to a Package

The minimum assurance level for the evaluation of a TOE conformant to this PP is EAL 2 augmented with ALC_FLR.2 and AVA_VAN_AP.3 as defined in section 6.3.

This conformance claim also applies to the PP-Configurations defined by combining the core MCU RoT PP with any subset of PP-Modules.

3.3 Conformance Claim of the PP

This PP does not claim conformance to any another PP.

3.4 Conformance Statement

The conformance to this PP, required for the STs and PPs claiming conformance to it, is strict, as defined in CC Part 1 [CC1].

This conformance claim also applies to the PP-Configurations defined by combining the core MCU RoT PP with any subset of PP-Modules.

3.5 Consistency Rationale for the PP-Modules

3.5.1 MCU RoT Persistent Time PP-Module

The MCU RoT Persistent Time PP-Module extends the core MCU RoT PP. It defines additional functionality, namely the monotonic ARoT persistent time.

The SPD of this PP-Module does not add any assumption or OSP and contains one new threat and the corresponding TOE’s security objective, both related to persistent time. There is no additional objective for the environment. The PP-Module adds three new SFRs for persistent time.

The unions of the SPD, the objectives, and the security functional requirements from the core MCU RoT PP and the PP-Module do not lead to a contradiction.

Furthermore, persistent time properties are independent from the functionalities defined in the MCU RoT ARoT Isolation PP-Module or MCU RoT Debug PP-Module. The three PP-Modules can be used independently or in combination.

3.5.2 MCU RoT Debug PP-Module

The MCU RoT Debug PP-Module extends the core MCU RoT PP. It defines additional functionality, namely the possibility for the MCU RoT Debug Administrator to be granted access to the Debug features upon authentication.

The SPD of this PP-Module does not add any assumption or OSP and contains one new threat and the corresponding TOE's security objective, both related to access control to the debug interface. There is no additional objective for the environment. The PP-Module adds nine new SFRs for access control.

The unions of the SPD, the objectives, and the security functional requirements from the core MCU RoT PP and the PP-Module do not lead to a contradiction.

Furthermore, the debug features are independent from the functionalities defined in the MCU RoT Persistent Time PP-Module or MCU RoT ARoT Isolation PP-Module. The three PP-Modules can be used independently or in combination.

3.5.3 MCU RoT ARoT Isolation PP-Module

The MCU RoT ARoT Isolation PP-Module extends the core MCU RoT PP. It defines additional functionality, namely it extends the isolation property defined in the core MCU RoT PP to isolation between Application Roots of Trust.

The SPD of this PP-Module does not add any assumption or OSP and contains one new threat and the corresponding TOE's security objective, both related to isolation between ARoTs. There is no additional objective for the environment. The PP-Module refines and replaces one SFR from the core MCU RoT PP.

The unions of the SPD, the objectives, and the security functional requirements from the core MCU RoT PP and the PP-Module do not lead to a contradiction.

Furthermore, ARoT isolation properties are independent from the functionalities defined in the MCU RoT Persistent Time PP-Module or MCU RoT Debug PP-Module. The three PP-Modules can be used independently or in combination.

4 Security Problem Definition

This chapter introduces the security problem addressed by the TOE, which consists of the threats a device enabled with an MCU RoT may face in the field, the assumptions on its operational environment, and the organisational security policies that must be implemented by the TOE or within the operational environment. The operational environment stands for the MCU RoT integration and maintenance environment, the ARoT development environment, and the usage environment of the devices enabled with an MCU RoT.

4.1 Assets

This section presents the assets of the TOE and their properties: authenticity, integrity, confidentiality, monotonicity, randomness, atomicity, read-only, and device binding (cf. section 1.4 for selected definitions).

4.1.1 Core MCU RoT PP

ARoT code

The code of the installed Application Roots of Trust. This data is typically stored in external non-volatile memory shared with the NSPE and potentially accessible by it.

Properties: Authenticity and integrity.

ARoT data and keys

Data and keys managed and stored by an ARoT using the MCU RoT security services. Data and keys are owned either by the user (the owner of the device enabled with the MCU RoT) or by the ARoT service provider. This data is typically stored in external non-volatile memory shared with NSPE and potentially accessible by it.

Properties: Authenticity, integrity, atomicity, confidentiality, and device binding.

ARoT instance time

Monotonic time during ARoT instance lifetime. Not affected by transitions through low power states. Not persistent over MCU RoT reset or MCU RoT shutdown.

Properties: Monotonicity.

Hardware Unique Key

The Hardware Unique Key (HUK) that is used to derive client and device specific keys, such as the key for trusted storage or attestation key. This data is typically stored in the Trusted OTP memory of the MCU.

Properties: Integrity and confidentiality.

Application Note:

Confidentiality of this asset is usually ensured by the simple fact that the asset remains inside the MCU.

MCU RoT firmware

The MCU RoT binary, containing MCU RoT code and constant data such as versioning information. This asset is typically stored in external non-volatile memory shared with the NSPE and potentially accessible by it.

Properties: Authenticity, integrity.

MCU RoT identifier

MCU RoT identification data that is globally unique among all MCU RoTs whatever the manufacturer, vendor, or integrator.

Properties: Unique and non-modifiable.

Application Note:

This data is typically stored in the Trusted OTP memory of the MCU or in a dedicated internal non-volatile storage. The MCU RoT identifier is intended to be public and exposed to any software running on the device, not only to Application RoTs. The TOE should return this identification through a standard Internal API.

MCU RoT initialisation code and data

Initialisation code and data (for instance, cryptographic certificates) used from device power-on up to the complete activation of the MCU RoT security services. Authentication of the MCU RoT is part of its initialisation.

Properties: Integrity.

MCU RoT persistent data

MCU RoT persistent data includes MCU RoT cryptographic keys (for instance, keys to authenticate ARoT code) and ARoT properties. This data may be stored in internal memory or in external non-volatile memory shared with the NSPE and potentially accessible by it.

Properties: Authenticity, integrity, confidentiality, and device binding.

MCU RoT rollback detection data

The MCU RoT data which is used to detect rollback of previous versions of trusted storage.

Properties: Integrity.

MCU RoT runtime data

MCU RoT runtime data includes execution variables, runtime context, random numbers generated by the MCU RoT, etc. This data is stored in volatile memory.

Properties: Integrity and confidentiality.

RNG

Random Number Generator.

Properties: Unpredictable random numbers, sufficient entropy, confidentiality, integrity.

4.1.2 MCU RoT Persistent Time PP-Module

The asset of this PP-Module extends the assets of the core MCU RoT PP as follows:

- “ARoT persistent time” is a new asset.

ARoT persistent time

Monotonic ARoT time between two time setting operations performed by any instance of the ARoT and persistent over MCU RoT reset.

Properties: Monotonicity.

4.1.3 MCU RoT Debug PP-Module

The asset of this PP-Module extends the assets of the core MCU RoT PP by providing a new cryptographic key.

MCU RoT Debug authentication key

The MCU RoT Debug authentication key used to authenticate the MCU RoT Debug Administrator for granting access to debug features.

Properties: Integrity and confidentiality.

4.1.4 MCU RoT ARoT Isolation PP-Module

This PP-Module does not introduce any new asset, nor does it extend any asset of the core MCU RoT PP.

4.2 Users

There are two kinds of users of the TOE: Application RoTs, which use the TOE services through an Internal API, and the Non-Secure Processing Environment, which uses the TOE’s services exported by the MCU RoT and the ARoTs.

4.2.1 Core MCU RoT PP

Application RoT (ARoT)

All the Application Roots of Trust running on the SPE are users of the TOE, through the MCU RoT Internal API.

Non-Secure Processing Environment (NSPE)

The Non-Secure Processing Environment, hosting the NSPE OS, the Client API, and the Client Applications that use the services of the MCU RoT and Application Roots of Trust, is a user of the TOE.

4.2.2 MCU RoT Persistent Time PP-Module

This PP-Module does not introduce any additional user.

4.2.3 MCU RoT Debug PP-Module

MCU RoT Debug Administrator

The MCU RoT Debug Administrator or actor acting on his behalf that can be granted access to MCU RoT Debug features.

4.2.4 MCU RoT ARoT Isolation PP-Module

This PP-Module does not introduce any additional user.

4.3 Threats

This Protection Profile targets threats to the TOE assets that arise during the end-usage phase and can be achieved by hardware or software means. Attacks with the most interest to the MCU RoT community are those that can be achieved by software means; are non-destructive; can be easily widespread, for instance through the internet. The chosen attack potential however does not exclude other means and types of attacks.

Attackers are individuals or organisations with remote or physical (local) access to the device embedding the MCU RoT. The user of the device becomes a potential attacker when the MCU RoT holds assets of third parties. The motivations behind the attacks may be very diverse and in general are linked to the Application Roots of Trust running on the MCU RoT. An attacker may, for instance, try to steal content of the device's owner (such as passwords stored in the device) or content of a service provider, or to unduly benefit from MCU RoT or ARoT services (such as network authentication), or to threaten the reputation of the device/MCU RoT manufacturer or service providers. The impact of an attack depends not only on the value of the individual assets attacked, but, in some cases, on the possibility to reproduce the attack rapidly at low cost: Single attacks performed on a given device enabled with an MCU RoT have a lower impact than massive attacks that reach many devices at the same time.

In many cases, a software attack involves at least two attackers: the attacker in the identification phase that discovers some hardware or software vulnerability, conceives malicious software to exploit it, and distributes that software; and the attacker at the exploitation phase that effectively exploits the vulnerability by running the malicious software (the end-user or a remote attacker on behalf of the user). The identification and the exploitation attackers may be the same person, in the case of an attack where there is no interest in or no possibility of spreading the attack widely.

Indeed, different device management and deployment models, as well as services, yield different expected threat models. For devices used in controlled environments, in which the installation of services is supervised, with the end-user having no value in breaking these services, the threat model addresses overall software attacks and vulnerabilities. An attack would indeed not be easily replicable as large-scale access to other such devices is not expected. For unmanaged, personal devices, an attack is more likely to be spreadable as, on the one hand, devices are more widespread and, on the other hand, the end-user himself may have interest in spreading the attack. Therefore, separation between identification and exploitation phases in an attack is key to evaluate such unmanaged devices.

Depending on the way the device is used, different assumptions may be valid regarding the identification phase in terms of means available to the attacker, software or hardware, and in terms of possibility to use more than one device, potentially in a destructive way. When identification and exploitation are separate, to address unmanaged devices across which an attack may be easily widespread, the attacker in the identification phase may have software and/or hardware expertise and access to equipment such as oscilloscopes, protocol analysers, in-circuit emulators, or JTAG debuggers, which allow the attacker to operate at the package interface on the PCB. However, the available attack potential is unlikely to be sufficient to act at deep package and SoC levels.

When identification and exploitation are separate, two main attacker profiles may arise in the exploitation phase:

- Remote attacker: This exploitation profile performs the attack on a remotely-controlled device or makes a downloadable tool that is very convenient to end-users. The attacker retrieves details of the vulnerability identified in the identification phase and produces such an attack code/executable. The attacker then makes a remote tool or malware and uses techniques such as phishing to have it downloaded and executed by a victim, or alternatively, makes a friendly tool available on the internet. Note that the design of a new malware, Trojan, virus, or rooting tool is often performed from an existing base, available on the internet
- Local layman attacker: This exploitation profile has physical access to the target device; the end-user or someone on his behalf may be the attacker. The attacker retrieves attack code/application from the identifier and/or guidelines written on the internet on how to perform the attack, and may download and use tools to root or reflash the device in order to get privileged access to the SPE allowing the execution of the exploit. These attacks do not require specialized equipment.

In all cases, the overall attack potential strongly limits the possibility to face advanced attackers performing the exploits.

The description of a threat provides its general goal; the threatened assets; and, in some cases, typical identification and/or exploitation attack paths. Some of the threats constitute in fact steps of longer attack paths related for instance to the disclosure or modification of assets. Nevertheless, they are stated separately to facilitate the tracing of the countermeasures.

4.3.1 Core MCU RoT PP

The following threats apply to any MCU RoT.

T.ABUSE_DEBUG

An attacker manages to be granted access to MCU RoT Debug features, allowing the attacker to obtain sensitive data or compromise the TSF (bypass, deactivate, or change security services).

Assets threatened directly: MCU RoT initialisation code and data (integrity), MCU RoT runtime data (confidentiality, integrity), ARoT code (authenticity, integrity), RNG (confidentiality, integrity).

Assets threatened indirectly: ARoT data and keys (confidentiality, authenticity, integrity) including instance time.

Application Note:

During the identification phase, the attacker may search for vulnerabilities for instance by exploiting the JTAG interface to access the MCU RoT Debug mode.

T.ABUSE_FUNCT

An attacker accesses MCU RoT functionality outside of its expected availability range, thus violating irreversible phases of the MCU RoT life cycle or state machine.

An attacker manages to instantiate an illegal MCU RoT or to start up the MCU RoT in an insecure state or to enter an insecure state, allowing the attacker to obtain sensitive data or to compromise the TSF (bypass, deactivate, or change security services).

Assets threatened directly: MCU RoT initialisation code and data (integrity), MCU RoT runtime data (confidentiality, integrity), ARoT code (authenticity, integrity), RNG (confidentiality, integrity).

Assets threatened indirectly: ARoT data and keys (confidentiality, authenticity, integrity) including instance time.

Application Note:

Attack paths may consist, for instance, in using commands in unexpected contexts or with unexpected parameters, impersonation of authorised entities, or exploiting badly implemented reset functionalities that provide undue privileges.

T.CLONE

An attacker manages to copy MCU RoT related data of a first device on a second device and makes this device accept them as genuine data.

Assets threatened directly: All data and keys (authenticity, device binding), MCU RoT identifier (authenticity, integrity).

T.FLASH_DUMP

An attacker partially or totally recovers the content of the external or internal Flash in cleartext, thus disclosing sensitive ARoT and MCU RoT data and potentially allowing the attacker to mount other attacks.

Assets threatened directly (confidentiality, authenticity, integrity): ARoT data and keys, MCU RoT persistent data.

Application Note:

An attack path consists for instance in performing a (partial) memory dump through the NSPE, purely via software.

During identification, another example consists in unsoldering the Flash memory and dumping its content, e.g. revealing a secret key that provides privileged access to many devices of the same model.

T.IMPERSONATION

An attacker impersonates an ARoT to gain unauthorised access to the services and data of another ARoT.

Assets threatened directly (confidentiality, integrity): MCU RoT runtime data, RNG.

Assets threatened indirectly: All data and keys (confidentiality, authenticity, integrity).

T.MCU_ROT_FIRMWARE_DOWNGRADE

An attacker backs up part or all of the MCU RoT firmware and restores it later in order to use obsolete MCU RoT functionality.

Assets threatened directly (integrity): MCU RoT firmware.

Assets threatened indirectly: All data and keys (confidentiality, authenticity, integrity).

T.PERTURBATION

An attacker modifies the behaviour of the MCU RoT or an ARoT in order to disclose or modify sensitive data or to force the MCU RoT or ARoT to execute unauthorised services.

Assets threatened directly: MCU RoT initialisation code and data (integrity), HUK (confidentiality, integrity), MCU RoT runtime data (confidentiality, integrity), RNG (confidentiality, integrity).

Assets threatened indirectly: All data and keys (confidentiality, authenticity, integrity) including ARoT instance time.

Application Note:

Unauthorised use of commands (one or many incorrect commands, undefined commands, hidden commands, invalid command sequence), buffer overflow attacks (overwriting buffer content to modify execution contexts or to gain system privileges) or fault injection (perturbation of the execution flow through voltage or clock glitches, for instance) are examples of attack paths. The MCU RoT can also be attacked through NSPE or ARoT "programmer errors" that, for example, exploit multi-threading, context/session management, or closed sessions; or by triggering system resets during execution of commands by the MCU RoT.

T.RAM

An attacker partially or totally recovers RAM content, thus disclosing runtime data and potentially allowing the attacker to interfere with the MCU RoT initialisation code and data.

Assets threatened directly: MCU RoT initialisation code and data (integrity), HUK (confidentiality, integrity), RoT runtime data (confidentiality, integrity), RNG (confidentiality, integrity).

Assets threatened indirectly: All data and keys (confidentiality, authenticity, integrity).

Application Note:

During the identification phase, an example attack path is to snoop on a memory bus, revealing code that is only decrypted at runtime, and finding a flaw in that code that can be exploited.

T.RNG

An attacker obtains information in an unauthorised manner about random numbers generated by the MCU RoT. This may occur, for instance, because the generated random numbers have insufficient entropy or because the attacker forces the output of a partially or totally predefined value.

Loss of unpredictability (the main property of random numbers) is a problem in case they are used to generate cryptographic keys. Malfunctions or premature ageing may also allow getting information about random numbers.

Assets threatened directly (confidentiality, integrity): RNG and secrets derived from random numbers.

T.ROGUE_CODE_EXECUTION

An attacker imports malicious code into the SPE to disclose or modify sensitive data.

Assets threatened directly (confidentiality, integrity): MCU RoT runtime data, RNG.

Assets threatened indirectly (confidentiality, authenticity, integrity): All assets.

Application Note:

Import of code within the NSPE is out of control of the MCU RoT.

T.ROLLBACK

An attacker backs up part or all storage spaces and restores them later in order to use obsolete ARoT services or to have the ARoT use obsolete data.

Assets threatened directly (confidentiality/integrity): ARoT data and keys, MCU RoT persistent data, ARoT code.

Assets threatened indirectly (confidentiality, integrity): MCU RoT runtime data, RNG.

Application Note:

Attacks may consist, for instance, in performing backup storage from Flash using the NSPE and restoring it later, or in modifying any MCU RoT persistent data used to detect a rollback.

T.SPY

An attacker discloses confidential data or keys by means of runtime attacks or unauthorised access to storage locations.

Assets threatened directly (confidentiality): All data and keys, HUK.

Application Note:

Exploitation of side-channels by a CA or ARoT (e.g. timing, power consumption), acquisition of residual sensitive data (e.g. improperly cleared memory), or use of undocumented or invalid command codes are examples of attack paths. The data may be used to exploit the device it was obtained on, or another device (e.g. a shared secret key).

During the identification phase, the attacker may for instance probe external buses.

T.STORAGE_CORRUPTION

An attacker corrupts all or part of the non-volatile storage used by the SPE including the trusted storage, in an attempt to trigger unexpected behaviour from the storage security mechanisms. The ultimate goal of the attack is to disclose and/or modify MCU RoT or ARoT data and/or code.

Assets threatened directly: HUK (confidentiality, integrity), MCU RoT persistent data (confidentiality, integrity), MCU RoT firmware (authenticity, integrity), ARoT data and keys (confidentiality, authenticity, integrity), ARoT instance time (imonotonicity), ARoT code (authenticity, integrity).

Application Note:

The attack can rely, for instance, on the NSPE file system or the Flash driver.

4.3.2 MCU RoT Persistent Time PP-Module

A new threat applies to MCU RoTs implementing ARoT persistent time.

T.AROT_PERSISTENT_TIME

An attacker modifies ARoT persistent time, for instance in order to extend expired rights or to produce fake logs.

Assets threatened directly (integrity): ARoT persistent time.

Assets threatened indirectly (confidentiality, integrity): ARoT data and keys.

Application Note:

Attacks may consist, for instance, in performing backup of the ARoT persistent time from Flash using the NSPE and restoring it later, in modifying the clock counter, or in removing the clock power supply.

4.3.3 MCU RoT Debug PP-Module

There is no additional threat in the MCU RoT Debug PP-Module.

4.3.4 MCU RoT ARoT Isolation PP-Module

A new threat applies to MCU RoTs implementing ARoT isolation.

T.ABUSE_AROT

An attacker accesses ARoT functionality outside of its expected availability range in order to obtain sensitive data or to compromise other ARoT data and keys.

Assets threatened directly: ARoT data and keys (confidentiality, authenticity, integrity).

4.4 Organisational Security Policies

This section presents the organisational security policies that have to be implemented by the TOE and/or its operational environment.

4.4.1 Core MCU RoT PP

The following policies apply to any MCU RoT.

OSP.CRYPTO_API

The TOE shall provide a well-defined set of cryptographic services to the Applications.

Application Note:

The ST author shall list all the cryptographic services and shall identify the services that are in the scope of the evaluation. Both sets may be empty.

OSP.INTEGRATION_CONFIGURATION

Integration and configuration of the MCU RoT by the device manufacturer shall rely on guidelines defined by the MCU RoT provider, which include all the security requirements issued from the TOE evaluation.

Application Note:

The ST author shall provide the reference of the MCU RoT guidelines, especially the operational guidance that fulfils AGD_OPE.1 requirements.

OSP.SECRETS

Generation, storage, distribution, destruction, injection of secret data in the MCU RoT or any other operation performed outside the MCU RoT shall enforce integrity and confidentiality of these data.

This applies to secret data injected before the end-usage phase (such as the HUK) or during the end-usage phase (such as cryptographic private or symmetric keys or any kind of confidential data).

OSP.UNIQUE_MCU_ROT_ID

The MCU RoT identifier, if not generated by the TOE but injected in the TOE, is globally unique among all MCU RoTs whatever the manufacturer, vendor, or integrator.

4.4.2 MCU RoT Persistent Time PP-Module

There is no additional organisational security policy in the MCU RoT Persistent Time PP-Module.

4.4.3 MCU RoT Debug PP-Module

The OSP.INTEGRATION_CONFIGURATION and OSP.SECRETS organisational security policies from the core MCU RoT PP also cover secure configuration of debug features and management of the secret used to authenticate the MCU RoT Debug Administrator.

There is no additional organisational security policy in the MCU RoT Debug PP-Module.

4.4.4 MCU RoT ARoT Isolation PP-Module

There is no additional organisational security policy in the MCU RoT ARoT Isolation PP-Module.

4.5 Assumptions

This section states the assumptions that apply to the TOE operational environment and its actors. Note that the operational environment is out of scope of the evaluation.

4.5.1 Core MCU RoT PP

The assumptions applicable to the core MCU RoT PP are the following:

A.AROT_DEVELOPMENT

ARoT developers are assumed to comply with the guidelines set by the MCU RoT provider. In particular, ARoT developers are assumed to consider the following principles:

- CA identifiers are generated and managed by the NSPE, outside the scope of the MCU RoT. An ARoT must not assume that CA identifiers are genuine.
- ARoTs must not disclose any sensitive data to the NSPE through any CA (interaction with the CA may require authentication means).
- Data written to memory that is not under the ARoT instance's exclusive control may have changed at next read.
- Reading twice from the same location in memory that is not under the ARoT instance's exclusive control can return different values.

A.PROTECTION_AFTER_DELIVERY

It is assumed that the TOE and its assets are protected by the operational environment after delivery (see section 2.5) and before entering the end-usage phase (phase 6 of the reference life cycle). It is assumed that the persons using the TOE in the operational environment understand and apply the MCU RoT guidelines (e.g. user and administrator guidance, installation documentation, personalization guide). It is also assumed that the persons responsible for the application of the procedures contained in the guidelines, and the persons involved in the delivery and protection of the product have the required skills and are aware of the security issues.

Application Note:

The TOE evaluation results and the related certificate are valid only when the guidelines are applied. For instance, for installation, pre-personalization, or personalization guides, only the described set-up configurations or personalization profiles are covered by the certificate.

4.5.2 MCU RoT Persistent Time PP-Module

There is no additional assumption in the MCU RoT Persistent Time PP-Module.

4.5.3 MCU RoT Debug PP-Module

There is no additional assumption in the MCU RoT Debug PP-Module.

4.5.4 MCU RoT ARoT Isolation PP-Module

There is no additional assumption in the MCU RoT ARoT Isolation PP-Module.

5 Security Objectives

5.1 Security Objectives for the TOE

This section states the security objectives for the MCU RoT. Since there is no mandatory split for the realisation of the security functions between software and hardware mechanisms, the objectives are close to the goal of the threats and allow any implementation.

5.1.1 Core MCU RoT PP

The following security objectives apply to any MCU RoT.

O.AROT_AUTHENTICITY

The TOE shall verify code authenticity of the binary code of the Application Roots of Trust.

Application Note:

Verification of authenticity of ARoT code can be performed together with the verification of MCU RoT firmware if both are bundled together or during loading of the code in volatile memory.

O.ATTESTATION

The TOE shall provide an initial attestation service to measure and attest the authenticity of the boot state of the MCU RoT, the security state of the MCU RoT, and the calling partition for the attestation service.

O.CA_AROT_IDENTIFICATION

The TOE shall provide means to protect the identity of each Application Root of Trust from usage by another resident Application Root of Trust and to distinguish Client Applications from Application Roots of Trust.

Application Note:

Client properties are managed either by the NSPE OS or by the SPM to ensure that a Client cannot tamper with its own properties in the following sense:

- The Client identity of an ARoT must always be determined by the SPM and the determination of whether it is an ARoT or not must be as trustworthy as the SPM itself.
- When the Client identity corresponds to an ARoT, then the SPM must ensure that the other Client properties are equal to the properties of the calling ARoT up to the same level of trustworthiness that the target ARoT places in the SPM.
- When the Client identity does not correspond to an ARoT, then the NSPE OS is responsible for ensuring that the Client Application cannot tamper with its own properties. However, this information is not trusted by the SPM.

O.INITIALISATION

The TOE shall be started through a secure initialisation process that ensures:

- The integrity of the MCU RoT initialisation code and data used to load the MCU RoT firmware
- The authenticity of the MCU RoT firmware
- The binding of the MCU RoT firmware to the SoC of the device
- Protection against MCU RoT firmware downgrade attacks.

Application Note:

Because the process is bound to the MCU, the HUK cannot be modified or tampered with.

O.INSTANCE_TIME

The TOE shall provide ARoT instance time and shall ensure that this time is monotonic during ARoT instance lifetime – from ARoT instance creation until the ARoT instance is destroyed – and not impacted by transitions through low power states.

O.KEYS_USAGE

The TOE shall enforce the cryptographic keys usage restrictions set by the creators of the keys.

O.MCU_ROT_DATA_PROTECTION

The TOE shall ensure the authenticity, integrity, and confidentiality of MCU RoT persistent data.

O.MCU_ROT_ID

The TOE shall ensure that the MCU RoT identifier is non-modifiable and provide means to retrieve this identifier. If generated by the TOE, it shall produce a universally unique identifier based on the RNG.

Application Note:

The MCU RoT identifier can be generated by the TOE (for instance using [RFC 4122]) or outside the TOE. When the MCU RoT identifier is generated outside the TOE, then before TOE delivery (in phase 3 or 5), and although it is not covered by this objective, there shall be an organisational process to ensure the uniqueness of the identifier.

O.MCU_ROT_ISOLATION

The TOE shall prevent the NSPE and the ARoTs from accessing the RoT's own execution and storage space and resources.

Application Note:

This objective contributes to the enforcement of the correct execution of the MCU RoT. It typically relies on hardware mechanisms. Note that resource allocation can change during runtime if it does not break isolation between MCU RoT resources during their usage and NSPE/ARoTs.

O.OPERATION

The TOE shall ensure the correct operation of its security functions. In particular, the TOE shall:

- Protect itself against abnormal situations caused by programmer errors or violation of good practices by the NSPE (and the CAs indirectly) or by the ARoTs.
- Control access to its services: the MCU RoT shall check the validity of any operation requested at any entry point into the MCU RoT.
- Enter a secure state upon failure detection, without exposure of any sensitive data.

Application Note:

- Programmer errors or violation of good practices (e.g. that exploit context/session management) might become attack-enablers. However, the MCU RoT must guarantee the stability and security of its resources and services independent of the NSPE, which may have been corrupted. In any case, an ARoT must not be able to use a programmer error on purpose to circumvent the security boundaries enforced by an implementation.
- Software in the NSPE must not be able to call directly to MCU RoT services. The NSPE software must go through protocols such that the SPM or an ARoT verifies the acceptability of the operation that the NSPE software has requested.

O.RNG

The TOE shall ensure the cryptographic quality of random number generation. Random numbers shall not be predictable and shall have sufficient entropy.

Application Note:

Random number generation may combine hardware and/or software mechanisms.

The RNG functionality is also used for generation of the MCU RoT identifier if this identifier is not generated outside the MCU RoT.

O.ROLLBACK_PROTECTION

The TOE shall prevent unauthorised rollback by:

- Monitoring for violation of the integrity of MCU RoT persistent data, MCU RoT rollback detection data, ARoT data or keys, or ARoT code
- Reacting to possible integrity violation so that security is always enforced.

Application Note:

This objective does not add any cryptographic measure to guarantee integrity or authenticity, since they are already required by O.RUNTIME_INTEGRITY, O.MCU_ROT_DATA_PROTECTION, and O.TRUSTED_STORAGE. However, this objective requires that the TSF must actively monitor potential integrity violations and take appropriate actions, should they happen.

O.RUNTIME_CONFIDENTIALITY

The TOE shall ensure that confidential MCU RoT runtime data and ARoT data and keys are protected against unauthorised disclosure. In particular:

- The TOE shall not export any sensitive data, random numbers, or secret keys to the NSPE.
- The TOE shall grant access to sensitive data, random numbers, or secret keys only to authorised ARoTs.

- The TOE shall clean up sensitive resources as soon as it can determine that their values are no longer needed.

O.RUNTIME_INTEGRITY

The TOE shall ensure that the MCU RoT firmware, MCU RoT runtime data, AROT code, and AROT data and keys are protected against unauthorised modification at runtime when stored in volatile memory.

O.TRUSTED_STORAGE

The TOE shall provide Trusted Storage services for persistent AROT general-purpose data and key material such that the following properties are enforced: confidentiality, authenticity, and integrity.

Moreover, the TOE shall enforce the atomicity of the operations that modify the storage or shall detect that modifications of the storage have not been completed as expected.

The MCU RoT Trusted Storage shall be bound to the host device, which means that the storage space must be accessible or modifiable only by authorised AROTs running on the same MCU RoT and device as when the data was created.

Table 7 summarizes which security objectives relate to the assets stored in non-volatile and/or volatile memory.

Table 7: Assets in Non-volatile or Volatile Memory

Asset	Security Objectives	
	Asset in Non-volatile Memory	Asset in Volatile Memory
AROT code	O.AROT_AUTHENTICITY	O.RUNTIME_INTEGRITY
AROT data and keys	O.TRUSTED_STORAGE	O.RUNTIME_INTEGRITY
Hardware Unique Key	O.INITIALISATION	O.RUNTIME_INTEGRITY
MCU RoT firmware	O.INITIALISATION	O.RUNTIME_INTEGRITY
MCU RoT identifier	O.MCU_ROT_ID	O.RUNTIME_INTEGRITY
MCU RoT initialisation code and data	O.INITIALISATION	O.RUNTIME_INTEGRITY
MCU RoT persistent data	O.MCU_ROT_DATA_PROTECTION	O.MCU_ROT_DATA_PROTECTION
MCU RoT rollback detection data	O.ROLLBACK_PROTECTION	O.RUNTIME_INTEGRITY
MCU RoT runtime data	NA	O.RUNTIME_INTEGRITY

5.1.2 MCU RoT Persistent Time PP-Module

The following objective applies to MCU RoTs implementing AROT persistent time.

O.AROT_PERSISTENT_TIME

The MCU RoT shall provide AROT persistent time, which is persistent over MCU RoT reset.

The MCU RoT shall ensure that:

- Either the persistent time is monotonic between two “time setting” operations performed by any instance of the ARoT,
- Or the persistent time is invalidated by detection of corruption.

5.1.3 MCU RoT Debug PP-Module

The following objective applies to MCU RoTs implementing debug features.

O.DEBUG

The MCU RoT shall authenticate the MCU RoT Debug Administrator before granting access to the MCU RoT Debug features.

5.1.4 MCU RoT ARoT Isolation PP-Module

The following objective applies to MCU RoTs implementing ARoT isolation.

O.AROT_ISOLATION

The MCU RoT shall isolate the ARoTs from each other: Each ARoT shall access its own execution and storage space, which is shared among all the instances of the ARoT but separated from the spaces of any other ARoT.

Application Note:

This objective contributes to enforcement of the confidentiality and integrity of the ARoT data.

5.2 Security Objectives for the Operational Environment

This section states the security objectives for the TOE operational environment covering all assumptions and organisational security policies that apply to the environment.

Application Note:

- The TOE operational environment is out of scope of the evaluation. The security guidelines must include recommendations to cover all the TOE objectives for the environment.
- The ST author shall provide the references of the applicable guidelines, particularly the preparative guidance that fulfils AGD_PRE.1 assurance requirements and the operational guidance that fulfils AGD_OPE.1 assurance requirements.

5.2.1 Core MCU RoT PP

The following security objectives apply to the TOE operational environment.

OE.AROT_DEVELOPMENT

ARoT developers shall comply with the ARoT development guidelines set by the MCU RoT provider. In particular, ARoT developers shall apply the following security recommendations during the development of the Application Roots of Trust:

- CA identifiers are generated and managed by the NSPE, outside the scope of the MCU RoT; therefore, ARoTs shall not assume that CA identifiers are genuine.
- ARoTs shall not disclose any sensitive data to the NSPE through any CA (interaction with the CA may require authentication means).
- ARoTs shall not assume that data written to a shared buffer can be read unchanged later on; ARoTs should always read data only once from the shared buffer and then validate it.
- ARoTs should copy the contents of shared buffers into ARoT instance-owned memory whenever these contents are required to be constant.

OE.DISABLED_DEBUG

All MCU RoT Debug interfaces shall be disabled in the end-usage phase.

OE.INTEGRATION_CONFIGURATION

Integration and configuration of the MCU RoT by the device manufacturer shall comply with the guidelines defined by the MCU RoT provider, which must include all the security requirements for the device manufacturer issued from the TOE evaluation.

OE.PROTECTION_AFTER_DELIVERY

The TOE and its assets shall be protected after delivery (see section 2.5) and before entering the end-usage phase (phase 6 of the reference life cycle). The personnel using the TOE in the operational environment shall apply the MCU RoT guidance (e.g. user and administrator guidance, installation documentation, personalization guide). The persons responsible for the application of the procedures contained in the guidelines, and the persons involved in the delivery and protection of the product have the required skills to understand and apply the guidelines and are aware of the security issues.

Application Note:

The certificate is valid only when the guidance is applied. For instance, for installation, pre-personalization, or personalization guides, only the described set-up configurations or personalization profiles are covered by the certificate.

OE.SECRETS

Management of secret data (e.g. generation, storage, distribution, destruction, loading into the product of cryptographic private keys, symmetric keys, user authentication data) performed outside the MCU RoT shall enforce integrity and confidentiality of these data.

OE.UNIQUE_MCU_ROT_ID

The MCU RoT identifier, if not generated by the TOE but injected in the TOE, shall be globally unique among all MCU RoTs whatever the manufacturer, vendor, or integrator.

5.2.2 MCU RoT Persistent Time PP-Module

There is no additional security objective for the Operational Environment in the MCU RoT Persistent Time PP-Module.

5.2.3 MCU RoT Debug PP-Module

There is no additional security objective for the Operational Environment in the MCU RoT Debug PP-Module.

Debug interface protection is enforced by the TOE (cf. O.DEBUG). Hence the objective for the operational environment OE.DISABLED_DEBUG from the core MCU RoT PP is discarded when the MCU RoT Debug PP-Module is used.

5.2.4 MCU RoT ARoT Isolation PP-Module

There is no additional security objective for the Operational Environment in the MCU RoT ARoT Isolation PP-Module.

5.3 Security Objectives Rationale

5.3.1 Threats

5.3.1.1 Core MCU RoT PP

T.ABUSE_DEBUG The objective for the environment OE.DISABLED_DEBUG ensures protection against abuse of debug functionality.

T.ABUSE_FUNCT The combination of the following objectives ensures protection against abuse of functionality:

- O.AROT_AUTHENTICITY ensures that the authenticity of ARoT code is verified.
- O.INITIALISATION ensures that the TOE security functionality is correctly initialised.
- O.KEYS_USAGE controls the usage of cryptographic keys.
- O.MCU_ROT_DATA_PROTECTION ensures that the data used by the MCU RoT are authentic and consistent.
- O.MCU_ROT_ISOLATION enforces the separation between the MCU RoT and the outside (NSPE and ARoTs).
- O.OPERATION ensures correct operation of the security functionality and proper management of failures.
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data.

- O.RUNTIME_INTEGRITY ensures protection against unauthorised modification of security functionality at runtime.
- OE.AROT_DEVELOPMENT enforces ARoT development principles, which are meant in particular to prevent disclosing information or performing modifications upon request of unauthorised entities.

T.CLONE The combination of the following objectives ensures protection against cloning:

- O.ATTESTATION ensures that external entities can verify the claimed identity and security parameters of the TOE.
- O.INITIALISATION ensures that the MCU RoT is bound to the SoC of the device.
- O.MCU_ROT_DATA_PROTECTION prevents the MCU RoT from using MCU RoT data that are inconsistent or not authentic.
- O.MCU_ROT_ID ensures that the MCU RoT identifier is non-modifiable.
- O.RNG ensures uniqueness of the MCU RoT identifier if generated inside the TOE.
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data, in particular TSF data used to bind the MCU RoT to the device.
- O.RUNTIME_INTEGRITY prevents unauthorised modification at runtime of security functionalities or data used to detect or prevent cloning.
- O.TRUSTED_STORAGE ensures that the trusted storage is bound to the device and prevents the RoT from using data that is inconsistent or not authentic.
- OE.UNIQUE_MCU_ROT_ID ensures uniqueness of the MCU RoT identifier if injected in the TOE.

T.FLASH_DUMP The objective O.TRUSTED_STORAGE ensures the confidentiality of the data stored in external memory.

T.IMPERSONATION The combination of the following objectives ensures protection against application impersonation attacks:

- O.CA_AROT_IDENTIFICATION ensures the protection of Client identities and the possibility to distinguish Client Applications and Application Roots of Trust.
- O.OPERATION ensures the verification of Client identities before any operation on their behalf.
- O.RUNTIME_INTEGRITY prevents unauthorised modification of security functionality at runtime.

T.MCU_ROT_FIRMWARE_DOWNGRADE The combination of the following objectives ensures protection against MCU RoT firmware downgrade:

- O.INITIALISATION ensures that the firmware that is executed is the intended version.
- OE.INTEGRATION_CONFIGURATION ensures that the firmware installed in the device is the intended version.
- OE.PROTECTION_AFTER_DELIVERY ensures that the firmware has not been modified after delivery.

T.PERTURBATION The combination of the following objectives ensures protection against perturbation attacks:

- O.AROT_AUTHENTICITY ensures that the authenticity of ARoT code is verified.
- O.INITIALISATION ensures that the TOE security functionality is correctly initialised.
- O.INSTANCE_TIME ensures the monotonicity of instance time stamps.
- O.MCU_ROT_DATA_PROTECTION covers MCU RoT persistent data which might influence the behaviour of the MCU RoT.
- O.MCU_ROT_ISOLATION enforces the separation between the MCU RoT and the outside (NSPE and ARoTs).
- O.OPERATION ensures correct operation of the security functionality and proper management of failures.
- O.RUNTIME_CONFIDENTIALITY covers MCU RoT runtime data which might influence the behaviour of the MCU RoT.
- O.RUNTIME_INTEGRITY ensures protection against unauthorised modification of security functionality at runtime.

Additional rationale specific to the MCU RoT Persistent Time PP-Module:

- O.AROT_PERSISTENT_TIME ensures the monotonicity of persistent time stamps.

Additional rationale specific to the MCU RoT ARoT Isolation PP-Module:

- O.AROT_ISOLATION ensures the separation of the ARoTs.

T.RAM The combination of the following objectives ensures protection against RAM attacks:

- O.INITIALISATION ensures that the TOE security functionality is correctly initialised and that the initialisation process is protected from the NSPE.
- O.MCU_ROT_ISOLATION provides a memory barrier between the MCU RoT, the ARoT layer and the NSPE.
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data at runtime.
- O.RUNTIME_INTEGRITY protects against unauthorised modification of code and data at runtime.

Additional rationale specific to the MCU RoT ARoT Isolation PP-Module:

- O.AROT_ISOLATION provides a memory barrier between ARoTs.

T.RNG The combination of the following objectives ensures protection of the random number generation:

- O.INITIALISATION ensures the correct initialisation of the TOE security functions, in particular the RNG.
- O.RNG ensures that random numbers are unpredictable, have sufficient entropy, and are not disclosed.
- O.RUNTIME_CONFIDENTIALITY ensures that confidential data is not disclosed.
- O.RUNTIME_INTEGRITY protects against unauthorised modification, for instance, against forcing the output of the RNG.

T.ROGUE_CODE_EXECUTION The combination of the following objectives ensures protection against import of malicious code:

- O.AROT_AUTHENTICITY ensures that the authenticity of ARoT code is verified.

- O.INITIALISATION ensures the integrity of MCU RoT firmware and that the TOE security functionality is correctly initialised.
- O.MCU_ROT_DATA_PROTECTION covers MCU RoT persistent data which might influence the behaviour of the RoT.
- O.OPERATION ensures correct operation of the security functionality.
- O.RUNTIME_CONFIDENTIALITY covers MCU RoT runtime data which might influence the behaviour of the MCU RoT.
- O.RUNTIME_INTEGRITY ensures protection against unauthorised modification of security functionality at runtime.
- O.TRUSTED_STORAGE ensures protection of the storage from which code might be imported.
- OE.INTEGRATION_CONFIGURATION covers the import of foreign code in a phase other than the end-usage phase.
- OE.PROTECTION_AFTER_DELIVERY covers the import of foreign code in a phase other than the end-usage phase.

T.ROLLBACK The objective O.ROLLBACK_PROTECTION ensures protection against rollback attacks.

T.SPY The combination of the following objectives ensures protection against disclosure:

- O.MCU_ROT_ISOLATION ensures that neither NSPE nor ARoTs can access MCU RoT data.
- O.RUNTIME_CONFIDENTIALITY ensures protection of confidential data at runtime.
- O.TRUSTED_STORAGE ensures that data stored in the trusted storage locations is accessible by the ARoT owner only.

Additional rationale specific to the MCU RoT ARoT Isolation PP-Module:

- O.AROT_ISOLATION ensures separation between the ARoTs.

T.STORAGE_CORRUPTION The combination of the following objectives ensures protection against corruption of non-volatile storage:

- O.AROT_AUTHENTICITY ensures that the authenticity of ARoT code is verified.
- O.INITIALISATION ensures that the firmware that is executed is the intended version.
- O.MCU_ROT_DATA_PROTECTION ensures that stored MCU RoT data are genuine and consistent.
- O.OPERATION ensures the correct operation of the TOE security functionality, including storage.
- O.ROLLBACK_PROTECTION ensures that the TSF monitors and handles integrity violations of the storage.
- O.TRUSTED_STORAGE enforces detection of corruption of the ARoT's storage.

5.3.1.2 MCU RoT Persistent Time PP-Module

T.AROT_PERSISTENT_TIME The objective O.AROT_PERSISTENT_TIME ensures the monotonicity of persistent time stamps and the failure management in case of modification detection.

5.3.1.3 MCU RoT Debug PP-Module

T.ABUSE_DEBUG The objective O.DEBUG ensures protection against abuse of debug functionality by authenticating the MCU RoT Debug Administrator before granting access to MCU RoT Debug features.

5.3.1.4 MCU RoT ARoT Isolation PP-Module

T.ABUSE_AROT The objective O.AROT_ISOLATION ensures protection against abuse of ARoT functionalities by ensuring isolation between Application Roots of Trust.

5.3.2 Organisational Security Policies

5.3.2.1 Core MCU RoT PP

OSP.INTEGRATION_CONFIGURATION The objective OE.INTEGRATION_CONFIGURATION directly covers this OSP.

OSP.SECRETS The objective OE.SECRETS directly covers this OSP.

OSP.UNIQUE_MCU_ROT_ID The objective OE.UNIQUE_MCU_ROT_ID directly covers this OSP.

5.3.3 Assumptions

5.3.3.1 Core MCU RoT PP

A.AROT_DEVELOPMENT The objective OE.AROT_DEVELOPMENT directly covers this assumption.

A.PROTECTION_AFTER_DELIVERY The objective OE.PROTECTION_AFTER_DELIVERY directly covers this assumption.

5.3.4 SPD and Security Objectives Rationale

Table 8 and Table 9 present the mapping between threats and security objectives.

Table 8: Threats and Security Objectives – Coverage

Threats	Security Objectives	Rationale
T.ABUSE_DEBUG	O.DEBUG (with Debug PP-Module) OE.DISABLED_DEBUG (without Debug PP-Module)	section 5.3.1
T.ABUSE_FUNCT	O.AROT_AUTHENTICITY O.INITIALISATION O.KEYS_USAGE O.MCU_ROT_DATA_PROTECTION O.MCU_ROT_ISOLATION O.OPERATION O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY OE.AROT_DEVELOPMENT	section 5.3.1
T.CLONE	O.ATTESTATION O.INITIALISATION O.MCU_ROT_DATA_PROTECTION O.MCU_ROT_ID O.RNG O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY O.TRUSTED_STORAGE OE.UNIQUE_MCU_ROT_ID	section 5.3.1
T.FLASH_DUMP	O.TRUSTED_STORAGE	section 5.3.1
T.IMPERSONATION	O.CA_AROT_IDENTIFICATION O.OPERATION O.RUNTIME_INTEGRITY	section 5.3.1
T.MCU_ROT_FIRMWARE_DOWNGRADE	O.INITIALISATION OE.INTEGRATION_CONFIGURATION OE.PROTECTION_AFTER_DELIVERY	section 5.3.1

Threats	Security Objectives	Rationale
T.PERTURBATION	O.AROT_AUTHENTICITY O.INITIALISATION O.INSTANCE_TIME O.MCU_ROT_DATA_PROTECTION O.MCU_ROT_ISOLATION O.OPERATION O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY and also O.AROT_ISOLATION (with ARoT Isolation PP-Module) O.AROT_PERSISTENT_TIME (with Persistent Time PP-Module)	section 5.3.1
T.RAM	O.INITIALISATION O.MCU_ROT_ISOLATION O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY and also O.AROT_ISOLATION (with ARoT Isolation PP-Module)	section 5.3.1
T.RNG	O.INITIALISATION O.RNG O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY	section 5.3.1
T.ROGUE_CODE_EXECUTION	O.AROT_AUTHENTICITY O.INITIALISATION O.MCU_ROT_DATA_PROTECTION O.OPERATION O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY O.TRUSTED_STORAGE OE.INTEGRATION_CONFIGURATION OE.PROTECTION_AFTER_DELIVERY	section 5.3.1
T.ROLLBACK	O.ROLLBACK_PROTECTION	section 5.3.1
T.SPY	O.MCU_ROT_ISOLATION O.RUNTIME_CONFIDENTIALITY O.TRUSTED_STORAGE and also O.AROT_ISOLATION (with ARoT Isolation PP-Module)	section 5.3.1

Threats	Security Objectives	Rationale
T.STORAGE_CORRUPTION	O.AROT_AUTHENTICITY O.INITIALISATION O.MCU_ROT_DATA_PROTECTION O.OPERATION O.ROLLBACK_PROTECTION O.TRUSTED_STORAGE	section 5.3.1
T.AROT_PERSISTENT_TIME (with Persistent Time PP-Module)	O.AROT_PERSISTENT_TIME (with Persistent Time PP-Module)	section 5.3.1
T.ABUSE_AROT (with ARoT Isolation PP-Module)	O.AROT_ISOLATION (with ARoT Isolation PP-Module)	section 5.3.1

Table 9: Security Objectives and Threats – Coverage

Security Objectives	Threats
O.AROT_AUTHENTICITY	T.ABUSE_FUNCT T.PERTURBATION T.ROGUE_CODE_EXECUTION T.STORAGE_CORRUPTION
O.ATTESTATION	T.CLONE
O.CA_AROT_IDENTIFICATION	T.IMPERSONATION
O.INITIALISATION	T.ABUSE_FUNCT T.CLONE T.MCU_ROT_FIRMWARE_DOWNGRADE T.PERTURBATION T.RAM T.RNG T.ROGUE_CODE_EXECUTION T.STORAGE_CORRUPTION
O.INSTANCE_TIME	T.PERTURBATION
O.KEYS_USAGE	T.ABUSE_FUNCT
O.MCU_ROT_DATA_PROTECTION	T.ABUSE_FUNCT T.CLONE T.PERTURBATION T.ROGUE_CODE_EXECUTION T.STORAGE_CORRUPTION
O.MCU_ROT_ID	T.CLONE

Security Objectives	Threats
O.MCU_ROT_ISOLATION	T.ABUSE_FUNCT T.PERTURBATION T.RAM T.SPY
O.OPERATION	T.ABUSE_FUNCT T.IMPERSONATION T.PERTURBATION T.ROGUE_CODE_EXECUTION T.STORAGE_CORRUPTION
O.RNG	T.CLONE T.RNG
O.ROLLBACK_PROTECTION	T.ROLLBACK T.STORAGE_CORRUPTION
O.RUNTIME_CONFIDENTIALITY	T.ABUSE_FUNCT T.CLONE T.PERTURBATION T.RAM T.RNG T.ROGUE_CODE_EXECUTION T.SPY
O.RUNTIME_INTEGRITY	T.ABUSE_FUNCT T.CLONE T.IMPERSONATION T.PERTURBATION T.RAM T.RNG T.ROGUE_CODE_EXECUTION
O.TRUSTED_STORAGE	T.CLONE T.FLASH_DUMP T.ROGUE_CODE_EXECUTION T.SPY T.STORAGE_CORRUPTION
O.AROT_ISOLATION (with ARoT Isolation PP-Module)	T.ABUSE_AROT T.PERTURBATION T.RAM T.SPY
O.AROT_PERSISTENT_TIME (with Persistent Time PP-Module)	T.AROT_PERSISTENT_TIME T.PERTURBATION
O.DEBUG (with Debug PP-Module)	T.ABUSE_DEBUG

Security Objectives	Threats
OE.AROT_DEVELOPMENT	T.ABUSE_FUNCT
OE.DISABLED_DEBUG (without Debug PP-Module)	T.ABUSE_DEBUG
OE.INTEGRATION_CONFIGURATION	T.MCU_ROT_FIRMWARE_DOWNGRADE T.ROGUE_CODE_EXECUTION
OE.PROTECTION_AFTER_DELIVERY	T.MCU_ROT_FIRMWARE_DOWNGRADE T.ROGUE_CODE_EXECUTION
OE.SECRETS	
OE.UNIQUE_MCU_ROT_ID	T.CLONE

Table 10 and Table 11 present the mapping between OSPs and security objectives.

Table 10: OSPs and Security Objectives – Coverage

Organisational Security Policies	Security Objectives	Rationale
OSP.INTEGRATION_CONFIGURATION	OE.INTEGRATION_CONFIGURATION	section 5.3.2
OSP.SECRETS	OE.SECRETS	section 5.3.2
OSP.UNIQUE_MCU_ROT_ID	OE.UNIQUE_MCU_ROT_ID	section 5.3.2

Table 11: Security Objectives and OSPs – Coverage

Security Objectives	Organisational Security Policies
O.AROT_AUTHENTICITY	
O.ATTESTATION	
O.CA_AROT_IDENTIFICATION	
O.INITIALISATION	
O.INSTANCE_TIME	
O.KEYS_USAGE	
O.MCU_ROT_DATA_PROTECTION	
O.MCU_ROT_ID	
O.MCU_ROT_ISOLATION	
O.OPERATION	
O.RNG	
O.ROLLBACK_PROTECTION	
O.RUNTIME_CONFIDENTIALITY	
O.RUNTIME_INTEGRITY	

Security Objectives	Organisational Security Policies
O.TRUSTED_STORAGE	
O.AROT_ISOLATION	
O.AROT_PERSISTENT_TIME (with Persistent Time PP-Module)	
O.DEBUG (with Debug PP-Module)	
OE.AROT_DEVELOPMENT	
OE.DISABLED_DEBUG (without Debug PP-Module)	
OE.INTEGRATION_CONFIGURATION	OSP.INTEGRATION_CONFIGURATION
OE.PROTECTION_AFTER_DELIVERY	
OE.SECRETS	OSP.SECRETS
OE.UNIQUE_MCU_ROT_ID	OSP.UNIQUE_MCU_ROT_ID

Table 12 and Table 13 present the mapping between assumptions and security objectives for the TOE operational environment.

Table 12: Assumptions and Security Objectives for the Operational Environment – Coverage

Assumptions	Security Objectives for the Operational Environment	Rationale
A.AROT_DEVELOPMENT	OE.AROT_DEVELOPMENT	section 5.3.3
A.PROTECTION_AFTER_DELIVERY	OE.PROTECTION_AFTER_DELIVERY	section 5.3.3

Table 13: Security Objectives for the Operational Environment and Assumptions – Coverage

Security Objectives for the Operational Environment	Assumptions
OE.AROT_DEVELOPMENT	A.AROT_DEVELOPMENT
OE.DISABLED_DEBUG (without Debug PP-Module)	
OE.INTEGRATION_CONFIGURATION	
OE.PROTECTION_AFTER_DELIVERY	A.PROTECTION_AFTER_DELIVERY
OE.SECRETS	
OE.UNIQUE_MCU_ROT_ID	

6 Extended Requirements

6.1 FCS_RNG – Random Numbers Generation

Family behaviour

To define the IT security functional requirements of the TOE, an additional family (FCS_RNG) of the Class FCS (cryptographic support) is defined here. This family describes the functional requirements for random number generation used for cryptographic purposes.

This family defines quality requirements for the generation of random numbers which are intended to be used for cryptographic purposes.

Component levelling

There is only one level in this family.

FCS_RNG.1 requires that random numbers meet a defined quality metric.

Management: FCS_RNG.1

No management activities are foreseen.

Audit: FCS_RNG.1

No actions are defined to be auditable.

FCS_RNG.1 Random numbers generation

Hierarchical to: No other components.

Dependencies: No dependencies.

FCS_RNG.1.1 The TSF shall provide a [selection: *physical, non-physical true, deterministic, hybrid, hybrid deterministic*] random number generator that implements: [assignment: *list of security capabilities*].

FCS_RNG.1.2 The TSF shall provide random numbers that meet [assignment: *a defined quality metric*].

6.2 FPT_INI – TSF Initialisation

Family behaviour

To define the security functional requirements of the TOE, an additional family (FPT_INI) of the Class FPT (Protection of the TSF) is introduced here. This family describes the functional requirements for the initialisation of the TSF by a dedicated function of the TOE that ensures the initialisation to a correct and secure operational state.

Component levelling

There is only one level in this family.

FPT_INI.1 requires the TOE to provide a TSF initialisation function that brings the TSF into a secure operational state at power-on.

Management: FPT_INI.1

No management activities are foreseen.

Audit: FPT_INI.1

No actions are defined to be auditable.

FPT_INI.1 TSF initialisation

Hierarchical to: No other components.

Dependencies: No dependencies.

FPT_INI.1.1 The TOE shall provide an initialisation function which is self-protected for integrity and authenticity.

FPT_INI.1.2 The TOE initialisation function shall ensure that certain properties hold on certain elements immediately before establishing the TSF in a secure initial state, as specified below:

ID	Properties	Elements
	[*assignment: property, for instance authenticity, integrity, correct version*]	[*assignment*: *list of TSF/user firmware, software or data*]

FPT_INI.1.3 The TOE initialisation function shall detect and respond to errors and failures during initialisation such that the TOE [selection: *is halted, successfully completes initialisation with [selection: reduced functionality, signalling error state, [assignment: list of actions]]*].

FPT_INI.1.4 The TOE initialisation function shall only interact with the TSF in [assignment: *defined methods*] during initialisation.

6.3 AVA_VAN_AP – Vulnerability Analysis

Objectives

Vulnerability analysis is an assessment to determine whether potential vulnerabilities identified in the TOE could allow attackers to violate the SFRs and thus to perform unauthorised access or modification to data or functionality.

The potential vulnerabilities may be identified either during the evaluation of the development, manufacturing, or assembly environments; during the evaluation of the TOE specifications, guidance, and available implementation representation; during anticipated operation of the TOE components; or by other methods, such as statistical methods.

The family 'Vulnerability analysis (AVA_VAN_AP)' defines requirements for evaluator independent vulnerability search and penetration testing of the TOE. Formally, AVA_VAN_AP extends the standard AVA_VAN.2 component by allowing the evaluator to require parts of the implementation representation and by performing penetration testing based on attack potential higher than Basic.

Note: Underlined text highlights the differences from AVA_VAN.2.

Component levelling

This Protection Profile defines one level of vulnerability analysis, namely AVA_VAN_AP.3 associated with Enhanced-basic attack potential.

AVA_VAN_AP.3 Vulnerability analysis

Dependencies:	ADV_ARC.1	Security architecture description
	ADV_FSP.2	Security-enforcing functional specification
	ADV_TDS.1	Basic design
	AGD_OPE.1	Operational user guidance
	AGD_PRE.1	Preparative procedures

Objectives

A vulnerability analysis is performed by the evaluator to ascertain the presence of potential vulnerabilities.

The evaluator performs penetration testing on the TOE to confirm that the potential vulnerabilities cannot be exploited in the operational environment. Penetration testing is performed by the evaluator assuming Enhanced-basic attack potential.

Developer action elements:

AVA_VAN_AP.3.1D The developer shall provide the TOE for testing.

Content and presentation elements:

AVA_VAN_AP.3.1C The TOE shall be suitable for testing.

Evaluator action elements:

AVA_VAN_AP.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN_AP.3.2E The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN_AP.3.3E The evaluator shall perform an independent vulnerability analysis of the TOE using the guidance documentation, functional specification, TOE design and security architecture description and the following parts of the TSF implementation representation: [selection: *none, [assignment: parts of the implementation representation*]] to identify potential vulnerabilities in the TOE.

AVA_VAN_AP.3.4E The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Enhanced-basic attack potential.

7 Security Requirements

This chapter defines the security functional and assurance requirements that apply to the TOE

7.1 Security Functional Requirements

This section provides the set of Security Functional Requirements (SFRs) for the TOE, which are derived from its security objectives.

7.1.1 Core MCU RoT PP

This Protection Profile uses the following security functional components, defined in CC Part 2 [CC2]:

- FAU_ARP.1 Security alarms
- FAU_SAR.1 Audit review
- FAU_STG.1 Protected audit trail storage
- FCO_NRO.1 Selective proof of origin
- FCS_COP.1 Cryptographic operation
- FDP_ACC.1 Subset access control
- FDP_ACF.1 Security attribute based access control
- FDP_IFC.2 Complete information flow control
- FDP_IFF.1 Simple security attributes
- FDP_ITT.1 Basic internal transfer protection
- FDP_RIP.1 Subset residual information protection
- FDP_ROL.1 Basic rollback
- FDP_SDI.2 Stored data integrity monitoring and action
- FIA_ATD.1 User attribute definition
- FIA_UID.2 User identification before any action
- FIA_USB.1 User-subject binding
- FMT_MSA.1 Management of security attributes
- FMT_MSA.3 Static attribute initialisation
- FMT_SMF.1 Specification of management functions
- FMT_SMR.1 Security roles
- FPT_FLS.1 Failure with preservation of secure state
- FPT_ITT.1 Basic internal TSF data transfer protection
- FPT_STM.1 Reliable time stamps

- FPT_TEE.1 Testing of external entities

Moreover, the following extended security functional components, defined in Chapter 6, are used:

- FCS_RNG.1 Random numbers generation
- FPT_INI.1 TSF initialisation.

The statement of the security functional requirements relies on the following characterisation of the MCU RoT in terms of users, subjects, objects, information, user data, TSF data, operations, and their security attributes (cf. CC Part 1 [CC1] for the definition of these notions).

User denotes an entity outside the TOE:

- Client Applications (CA), with security attribute "CA_identity" (CA identifier)
- Application Roots of Trust (ARoTs), with security attributes "ARoT_identity" (ARoT identifier), "ARoT_properties".

Subject denotes an active entity inside the TOE:

- S.API: The MCU RoT Internal API, with security attributes "caller" (ARoT identifier)
- S.RESOURCE: Any software or hardware component which may be used alternatively by the MCU RoT or the NSPE, with security attribute "state" (RoT/NSPE). E.g. cryptographic accelerator, random number generator, cache, registers. Note: When the state is NSPE, the MCU RoT may access the resource. The communication buses are not considered subjects (cf. FDP_ITT.1).
- S.RAM_UNIT: RAM addressable unit, with security attribute "rights:(ARoT identifier/NSPE) -> (Read/Write/ReadWrite/NoAccess). For instance, an addressable unit may be allocated or have its access rights changed upon ARoT instance creation or when sharing memory references between a client (CA, ARoT) and an ARoT. Notes: 1) A RAM_UNIT typically stands for a byte in the C language; 2) There is no RAM access restriction applicable to the RoT itself.
- S.COMM_AGENT: Proxy between CAs in the NSPE and the MCU RoT and its ARoTs.
- S.AROT_INSTANCE: Any ARoT instance with security attribute "ARoT_identity" (ARoT identifier).
- S.AROT_INSTANCE_SESSION: Any session within a given ARoT instance, with security attribute "client_identity" (CA identifier).

Object denotes a passive entity inside the TOE:

- OB.AROT_STORAGE (user data): Trusted Storage space of an ARoT, with security attributes "owner" (ARoT identifier), "inExtMem" (True/False), and "RoT_identity" (MCU RoT identifier)
- OB.HUK (TSF data): The HUK, with security attribute "RoT_identity" (MCU RoT identifier).

A *cryptographic object* is a special kind of RoT object:

- OB.AROT_KEY (user data): (handle to a) user key (persistent or transient), with security attributes "usage", "owner" (ARoT identifier), "isExtractable" (True/False).

Information denotes data exchanged between subjects:

- I.RUNTIME_DATA (user data or TSF Data depending on the owner): Data belonging to the ARoT or to the RoT itself. May include parameter values, return values, content of memory regions in cleartext. Note: Data that is encrypted and authenticated is not considered I.RUNTIME_DATA.

TSF data consists of runtime TSF data and persistent TSF data (also called MCU RoT persistent data) necessary to provide the security services. It includes all the security attributes of users, subjects, objects, and information.

Operation denotes

- *A cryptographic operation* on user keys performed by S.API on behalf of S.AROT_INSTANCE:
  - OP.USE_KEY: Any cryptographic operation that uses a key
  - OP.EXTRACT_KEY: Any operation that populates a key.
- *A Trusted Storage operation* performed by S.API on behalf of S.AROT_INSTANCE:
  - OP.LOAD: Any operation used to get back persistent objects (data and keys) to be used by the ARoT
  - OP.STORE: Any operation used to store persistent objects (data and keys): object creation, object deletion, object renaming, object truncation, and write to an object.
- Any other operation executed by the RoT on behalf of S.AROT_INSTANCE.

This PP defines the following access control and information flow security functional policies (SFP):

Runtime Data Information Flow Control SFP:

- Purpose: To control the flow of runtime data from and to executable entities and memory. This policy contributes to ensuring the integrity and confidentiality of runtime data.
- Subjects: S.AROT_INSTANCE, S.AROT_INSTANCE_SESSION, S.API, S.COMM_AGENT, S.RESOURCE, S.RAM_UNIT
- Information: I.RUNTIME_DATA
- Security attributes: S.RESOURCE.state, S.RAM_UNIT.rights, and S.API.caller
- Operations: all
- SFR instances: FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_ITT.1/Runtime.

ARoT Keys Access Control SFP:

- Purpose: To control access to ARoT keys, which is granted to the ARoT that owns the key only. This policy contributes to the confidentiality of ARoT keys.
- Subjects: S.API, S.AROT_INSTANCE, and any other subject in the MCU RoT
- Objects: OB.AROT_KEY
- Security attributes: OB.AROT_KEY.usage, OB.AROT_KEY.owner, OB.AROT_KEY.isExtractable, and S.API.caller
- Operations: OP.USE_KEY, OP.EXTRACT_KEY

- SFR instances: FDP_ACC.1/AROT_Keys, FDP_ACF.1/AROT_keys, FMT_MSA.1/AROT_keys, FMT_MSA.3/AROT_keys and FMT_SMF.1.

Trusted Storage Access Control SFP:

- Purpose: To control the access to ARoT storage where persistent ARoT data and keys are stored, which is granted on behalf of the owner ARoT only. This policy also enforces the binding of ARoT trusted storage to the HUK OB.HUK.
- Subjects: S.API
- Objects: OB.AROT_STORAGE, OB.HUK
- Security attributes: S.API.caller, OB.AROT_STORAGE.owner, OB.AROT_STORAGE.inExtMem, OB.AROT_STORAGE.ROT_identity, and OB.HUK.ROT_identity
- Operations: OP.LOAD, OP.STORE
- SFR instances: FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage and FMT_SMF.1.

Application Note:

The ST author shall fill in the SFR open assignments and perform the selections that are appropriate for their product. The TOE Summary Specification (TSS) shall describe how the product implements the instantiated requirements. Note that the requirements may imply supporting security functionality, for instance:

- Authentication/signature and encryption/decryption of storage spaces located in external memory (cf. FDP_ACF.1/Trusted Storage)
- Binding of Client Applications with ARoT sessions (cf. FIA_USB.1)
- Verification of the client_identity when the client is an ARoT (cf. FIA_USB.1)
- Binding of trusted storage with the HUK OB.HUK (cf. FDP_ACF.1/Trusted Storage)
- Configuration of MCU RoT resources shared with the NSPE (cf. FDP_IFF.1/Runtime)
- Secure state definition and entering process upon failure management (cf. FPT_FLS.1).

7.1.1.1 Identification

7.1.1.1.1 FIA_ATD.1 User attribute definition

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users: **CA_identity**, **ARoT_identity**, **ARoT_properties**, **[assignment: list of security attributes]**.

Application Note:

The lifetime of the attributes in such a list is the following:

- CA_identity: The lifetime of this attribute is that of the lifetime of the client session to the ARoT.
- ARoT_identity: The availability of this attribute is that of the availability of the ARoT to clients.
- ARoT_properties: The lifetime of this attribute is that of the availability of the ARoT to clients.

7.1.1.1.2 FIA_UID.2 User identification before any action

FIA_UID.2.1 The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

Application Note:

User stands for Client Application or Application Root of Trust.

7.1.1.1.3 FIA_USB.1 User-subject binding

FIA_USB.1.1 The TSF shall associate the following user security attributes with subjects acting on the behalf of that user:

- **Client (CA or ARoT) identity is codified into the client_identity of the requested ARoT session.**
- **[assignment: list of user security attributes].**

FIA_USB.1.2 The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users:

- **If the client is an ARoT, then the client_identity must be equal to the ARoT_identity of the ARoT subject that is the client.**
- **[assignment: rules for the initial association of attributes].**

FIA_USB.1.3 The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users:

- **No modification of client_identity is allowed after initialisation.**
- **[assignment: rules for the changing of attributes].**

7.1.1.1.4 FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles:

- **TSF**
- **ARoT_User**
- **[assignment: the authorised identified roles].**

FMT_SMR.1.2 The TSF shall be able to associate users with roles.

Application Note:

- The ARoT_User role is the TSF running on behalf of an ARoT, upon request from the NSPE (by Client Applications) or from other ARoT.

- The third item in FMT_SMR.1.1 leaves open the possibility of defining additional roles. However, the definition of such additional roles is not mandatory.
- The ST author shall define any additional role that is required by corresponding new rules should be included in FMT_MSA.1.

7.1.1.2 Confidentiality, Integrity, and Isolation

7.1.1.2.1 FDP_IFC.2/Runtime Complete information flow control

FDP_IFC.2.1/Runtime The TSF shall enforce the **Runtime Data Information Flow Control SFP** on:

- **Subjects: S.AROT_INSTANCE, S.AROT_INSTANCE_SESSION, S.API, S.COMM_AGENT, S.RESOURCE, S.RAM_UNIT**
- **Information: I.RUNTIME_DATA**

and all operations that cause that information to flow to and from subjects covered by the SFP.

FDP_IFC.2.2/Runtime The TSF shall ensure that all operations that cause any information in the TOE to flow to and from any subject in the TOE are covered by an information flow control SFP.

Application Note:

The flow control policy specifies the conditions to communicate runtime data from one subject to another. It applies to operations that are standard interfaces of these subjects.

7.1.1.2.2 FDP_IFF.1/Runtime Simple security attributes

FDP_IFF.1.1/Runtime The TSF shall enforce the **Runtime Data Information Flow Control SFP** based on the following types of subject and information security attributes: **S.RESOURCE.state**, **S.RAM_UNIT.rights**, and **S.API.caller**.

FDP_IFF.1.2/Runtime The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

- **Rules for information flow between S.AROT_INSTANCE and S.RAM_UNIT:**
  - **Flow of I.RUNTIME_DATA from S.AROT_INSTANCE to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(SPE) is Write or ReadWrite**
  - **Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.AROT_INSTANCE is allowed only if S.RAM_UNIT.rights(SPE) is Read or ReadWrite**
- **Rules for information flow from and to S.COMM_AGENT:**
  - **Flow of I.RUNTIME_DATA from S.COMM_AGENT to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(NSPE) is Write or ReadWrite**
  - **Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.COMM_AGENT is allowed only if S.RAM_UNIT.rights(NSPE) is Read or ReadWrite**

- **Rules for information flow from and to S.API:**
  - **Flow of I.RUNTIME_DATA from S.API to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(S.API.caller) is Write or ReadWrite**
  - **Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.API is allowed only if S.RAM_UNIT.rights(S.API.caller) is Read or ReadWrite**
- **Rules for information flow from and to S.RESOURCE:**
  - **Flow of I.RUNTIME_DATA between S.API and S.RESOURCE is allowed only if the resource is under MCU RoT control (S.RESOURCE.state = RoT).**

FDP_IFF.1.3/Runtime The TSF shall enforce the **[assignment: additional information flow control SFP rules]**.

FDP_IFF.1.4/Runtime The TSF shall explicitly authorise an information flow based on the following rules:

- **Rules for information flow from and to S.AROT_INSTANCE_SESSION:**
  - **Flow of I.RUNTIME_DATA that are parameter or return values is allowed between S.AROT_INSTANCE_SESSION and S.COMM_AGENT.**
  - **Flow of I.RUNTIME_DATA that are parameter or return values is allowed between S.AROT_INSTANCE_SESSION and S.API.**

FDP_IFF.1.5/Runtime The TSF shall explicitly deny an information flow based on the following rules: **Any information flow involving a MCU RoT subject unless one of the conditions stated in FDP_IFF.1.1/Runtime, FDP_IFF.1.2/Runtime, FDP_IFF.1.3/Runtime, FDP_IFF 1.4/Runtime holds**.

Application Note:

- The access rights configuration managed by S.RAM_UNIT ensures that RAM addressable units used for TSF data are appropriately protected (in integrity for MCU RoT firmware, in integrity and confidentiality for MCU RoT runtime data).
- RAM units can span over several volatile memories; for example, on-chip RAM, off-chip RAM, registers.
- MCU RoT-dedicated RAM units may hold copies of the content of temporary memory references passed by the NSPE.

7.1.1.2.3 FDP_ITT.1/Runtime Basic internal transfer protection

FDP_ITT.1.1/Runtime The TSF shall enforce the **Runtime Data Information Flow Control SFP** to prevent the **disclosure and modification** of user data when it is transmitted between physically-separated parts of the TOE.

Application Note:

The resources used by the MCU RoT may reside in “physically separated parts”. This requirement addresses data transmission through communication buses (recall that the definition of S.RESOURCES does not include

the buses).

7.1.1.2.4 FDP_RIP.1/Runtime Subset residual information protection

FDP_RIP.1.1/Runtime The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: MCU **RoT and ARoT runtime objects**.

Application Note:

This operation applies upon:

- Failure detection (cf. FPT_FLS.1)
- ARoT instance or session closing.

7.1.1.2.5 FPT_ITT.1/Runtime Basic internal TSF data transfer protection

FPT_ITT.1.1/Runtime The TSF shall protect TSF data from **disclosure and modification** when it is transmitted between separate parts of the TOE.

Application Note:

The resources used by the MCU RoT may reside in “physically separated parts”.

7.1.1.3 Cryptography

7.1.1.3.1 FCS_COP.1/API Cryptographic operation

FCS_COP.1.1/API The TSF shall perform **[assignment: list of cryptographic operations provided through MCU RoT Internal APIs]** in accordance with a specified cryptographic algorithm **[assignment: cryptographic algorithms]** and cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignment: list of standards]**.

Application Note:

The ST author shall include this SFR whenever the TSF provides cryptographic operations to ARoTs that are in the scope of the evaluation.

7.1.1.3.2 FCS_COP.1/Internals Cryptographic operation

FCS_COP.1.1/Internals The TSF shall perform **[assignment: list of cryptographic operations for the verification of the authenticity of MCU RoT firmware and ARoT code, for the integrity and the confidentiality of Trusted Storage, and for any other TSF internal function]** in accordance with a specified cryptographic algorithm **[assignment: cryptographic algorithm]** and cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignment: [CRec] and list of applicable standards]**.

Application Note:

The ST author shall specify the cryptographic operations used within the TOE for:

- Verifying the authenticity of MCU RoT firmware and ARoT code.
- Protecting the integrity and confidentiality of Trusted Storage data. These operations are based on the HUK.
- Performing measurements for software components and signing attestation reports.

7.1.1.3.3 FDP_ACC.1/ARoT_keys Subset access control

FDP_ACC.1.1/ARoT_keys The TSF shall enforce the **ARoT Keys Access Control SFP** on:

- **Subjects: S.API, S.AROT_INSTANCE, and any other subject in the MCU RoT**
- **Objects: OB.AROT_KEY**
- **Operations: OP.USE_KEY, OP.EXTRACT_KEY**

7.1.1.3.4 FDP_ACF.1/ARoT_keys Security attribute based access control

FDP_ACF.1.1/ARoT_keys The TSF shall enforce the **ARoT Keys Access Control SFP** to objects based on the following: **OB.AROT_KEY.usage**, **OB.AROT_KEY.owner**, **OB.AROT_KEY.isExtractable**, and **S.API.caller**.

FDP_ACF.1.2/ARoT_keys The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **OP.USE_KEY is allowed if the following conditions hold:**
  - **The ARoT instance that requested the operation to the API owns the key (S.API.caller = OB.AROT_KEY.owner).**
  - **The intended usage of the key (OB.AROT_KEY.usage) matches the requested operation.**
- **OP.EXTRACT_KEY is allowed if the following conditions hold:**
  - **The ARoT instance that requested the operation to the API owns the key (S.API.caller = OB.AROT_KEY.owner).**
  - **The operation attempts to extract the public part of OB.AROT_KEY or the key is extractable (OB.AROT_KEY.isExtractable = True).**

FDP_ACF.1.3/ARoT_keys The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly authorise access of subjects to objects]**.

FDP_ACF.1.4/ARoT_keys The TSF shall explicitly deny access of subjects to objects based on the following additional rules:

- **Any access to a user key attempted directly from S.AROT_INSTANCE or any other subject of the MCU RoT that is not S.API**

- **Any access to a user key attempted from S.API without valid caller (S.API.caller is undefined)**
- **[assignment: rules, based on security attributes, that explicitly deny access of subjects to objects].**

Application Note:

OP.USE_KEY and OP.EXTRACT_KEY stand for operations of the MCU RoT Internal API.

7.1.1.3.5 FMT_MSA.1/ARoT_keys Management of security attributes

FMT_MSA.1.1/ARoT_keys The TSF shall enforce the **ARoT Keys Access Control SFP** to restrict the ability to **change_default**, **query**, and **modify** the security attributes **OB.AROT_KEY.usage**, **OB.AROT_KEY.isExtractable, and OB.AROT_KEY.owner** to **the following roles**:

	change-default	**query**	**modify**
OB.AROT_KEY.usage	**ARoT_User**	**ARoT_User**	**ARoT_User**
OB.AROT_KEY.owner	**no role**	**TSF**	**no role**
OB.AROT_KEY.isExtractable	**no role**	**TSF**	**no role**

7.1.1.3.6 FMT_MSA.3/ARoT_keys Static attribute initialisation

FMT_MSA.3.1/ARoT_keys The TSF shall enforce the **ARoT Keys Access Control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/ARoT_keys The TSF shall allow the **ARoT_User role**, **[assignment: the authorised identified roles]** to specify alternative initial values to override the default values when an object or information is created.

7.1.1.4 Initialisation, Operation, and Firmware Integrity

7.1.1.4.1 FAU_ARP.1 Security alarms

FAU_ARP.1.1 The TSF shall take **[assignment: list of actions]** upon detection of a potential security violation.

Refinement:

The TSF shall take the following actions upon detection of a potential security violation:

- Detection of integrity violation of ARoT data, ARoT code, or MCU RoT data: **[assignment: associated actions]**
- Detection of MCU RoT firmware integrity violation: **[assignment: associated actions]**

- **[assignment: list of implementation-dependent potential security violations and associated actions]**

7.1.1.4.2 FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for **[assignment: integrity errors]** on all objects, based on the following attributes: **[assignment: user data attributes]**.

Refinement:

The TSF shall monitor MCU RoT runtime data, MCU RoT persistent data, ARoT data and keys, and ARoT code stored in containers controlled by the TSF for **authenticity and integrity errors** on all objects, based on the following attributes: **[assignment: attributes of MCU RoT runtime data, MCU RoT persistent data, ARoT data and keys, and ARoT code]**.

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **[assignment: action to be taken]**.

Refinement:

- Upon detection of authenticity or integrity errors in MCU RoT runtime data or MCU RoT persistent data, the TSF shall **[assignment: action that does not depend on the compromised data]**.
- Upon detection of authenticity or integrity errors in ARoT code, the TSF shall **abort the execution of the ARoT instance**.
- Upon detection of authenticity or integrity errors in ARoT data or ARoT keys, the TSF shall:
  - **Not give back any compromised data**
  - **[assignment: action that does not depend on the compromised data]**
- **[assignment: other actions to be taken].**

Application Note:

This SFR is used for both TSF and user data as similar mechanisms are involved to protect the integrity of this data. Rollback detection is ensured by rollback detection data and by integrity failure detection.

7.1.1.4.3 FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions:

- **Management of ARoT keys security attributes**
- **Provision of Trusted Storage security attributes to authorised users.**

7.1.1.4.4 FPT_FLS.1 Failure with preservation of secure state

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur:

- **Device binding failure**
- **Cryptographic operation failure**

- **Invalid CA requests, in particular bad-formed requests**
- **Panic states**
- **Authenticity or integrity failure of ARoT code, ARoT data or keys**
- **MCU RoT persistent data authenticity or integrity failure**
- **MCU RoT firmware integrity failure**
- **MCU RoT initialisation failure**
- **Unexpected commands in the current MCU RoT state**
- **[assignment: list of additional types of failures that may affect the TSF].**

Application Note:

- Device binding failure occurs when (part of) the stored data has not been bound by the same MCU RoT.
- The ST author shall define the characteristics of the secure state. In particular, the transition between a failure state and the secure state shall protect MCU RoT and user data and keys confidentiality.

7.1.1.4.5 FPT_INI.1 TSF initialisation

FPT_INI.1.1 The TOE shall provide an initialisation function which is self-protected for integrity and authenticity.

FPT_INI.1.2 The TOE initialisation function shall ensure that certain properties hold on certain elements immediately before establishing the TSF in a secure initial state, as specified below:

	Properties	Elements
1	**authenticity and integrity**	**MCU RoT firmware**
2	**prevention of downgrade to previous versions**	**MCU RoT firmware**
3	**integrity**	**HUK** **MCU RoT identifier** **MCU RoT initialisation code and data**
…	**[assignment: list of implementation-dependent verifications]**	**[assignment: list of firmware, software, or data]**

FPT_INI.1.3 The TOE initialisation function shall detect and respond to errors and failures during initialisation such that the TOE **[selection: is halted, successfully completes initialisation with [selection: reduced functionality, signalling error state, [assignment: list of actions]]**.

FPT_INI.1.4 The TOE initialisation function shall only interact with the TSF in **[assignment: defined methods]** during initialisation.

Application Note:

- FPT_INI.1.1 covers, for instance, code/data that are stored and executed from non-modifiable memory at boot time, the immutable root-of-trust, and other OTP values such as versions and identifiers.
- In FPT_INI.1.2, firmware downgrade verification must rely on data residing on the TOE, for instance on One Time Programmable (OTP) memories or EEPROM.

7.1.1.4.6 FPT_TEE.1 Testing of external entities

FPT_TEE.1.1 The TSF shall run a suite of tests **prior to execution and [assignment: other conditions]** to check **the fulfilment of authenticity of ARoT code**.

FPT_TEE.1.2 If the test fails, the TSF shall **not start the execution of the ARoT instance**.

7.1.1.5 MCU RoT Identification

7.1.1.5.1 FAU_SAR.1 Audit review

FAU_SAR.1.1 The TSF shall provide **all users** with the capability to read **the MCU RoT identifier** from the audit records.

FAU_SAR.1.2 The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

7.1.1.5.2 FAU_STG.1 Protected audit trail storage

FAU_STG.1.1 The TSF shall protect the stored audit records in the audit trail from unauthorised deletion.

FAU_STG.1.2 The TSF shall be able to **prevent** unauthorised modifications to the stored audit records in the audit trail.

Application Note:

- The audit record in this SFR refers to the MCU RoT identifier. This unique identifier is stored on the TOE before TOE delivery. It can be generated on or off the TOE (the ST author shall specify the generation method) and is not modifiable during the end-usage phase.
- The ST author shall indicate in which type of persistent memory the identifier is stored.

7.1.1.6 Instance Time

7.1.1.6.1 FPT_STM.1/Instance time Reliable time stamps

FPT_STM.1.1/Instance time The TSF shall be able to provide reliable time stamps.

Refinement:

The TSF shall be able to provide time stamps to ARoT instances such that time stamps are monotonic during the ARoT instance lifetime.

Application Note:

The refinement provides the meaning of the reliability that is expected.

7.1.1.7 Random Number Generator

7.1.1.7.1 FCS_RNG.1 Random numbers generation

FCS_RNG.1.1 The TSF shall provide *a* **[selection: physical, non-physical true, deterministic, hybrid, hybrid deterministic]** random number generator that implements: **[assignment: list of security capabilities]**.

FCS_RNG.1.2 The TSF shall provide random numbers that meet **[assignment: a defined quality metric].**

Application Note:

- The terms physical, non-physical true, deterministic, hybrid, and hybrid deterministic RNGs are defined in [AIS31].
- The ST author shall complete the elements FCS_RNG.1.1 and FCS_RNG_1.2. The ST author should define the quality of the generated random numbers using for instance the Min-entropy or Shannon entropy. The assignment of a quality metric shall ensure sufficient randomness of the random numbers near to the uniform distributed random variables. The evaluation of the random number generator shall follow a recognized methodology, e.g. [AIS31] or [SP800-90]. This SFR is also used to ensure uniqueness of the MCU RoT identifier if it is generated on the TOE.

7.1.1.8 Trusted Storage

7.1.1.8.1 FDP_ACC.1/Trusted Storage Subset access control

FDP_ACC.1.1/Trusted Storage The TSF shall enforce the **Trusted Storage Access Control SFP** on:

- **Subjects: S.API**
- **Objects: OB.AROT_STORAGE, OB.HUK**
- **Operations: OP.LOAD, OP.STORE.**

7.1.1.8.2 FDP_ACF.1/Trusted Storage Security attribute based access control

FDP_ACF.1.1/Trusted Storage The TSF shall enforce the **Trusted Storage Access Control SFP** to objects based on the following: **S.API.caller**, **OB.AROT_STORAGE.owner**, **OB.AROT_STORAGE.inExtMem**, **OB.AROT_STORAGE.ROT_identity**, and **OB.HUK.ROT_identity**.

FDP_ACF.1.2/Trusted Storage The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **OP.LOAD of an object from OB.AROT_STORAGE is allowed if the following conditions hold:**
  - **The operation is performed by S.API.**
  - **The load request comes from an instance of the owner of the trusted storage space (S.API.caller = OB.AROT_STORAGE.owner).**
  - **OB.AROT_STORAGE is bound to the HUK OB.HUK (OB.AROT_STORAGE.ROT_identity = OB.HUK.ROT_identity).**
  - **If OB.AROT_STORAGE is located in external memory accessible to the NSPE (OB.AROT_STORAGE.inExtMem = True), then the object is authenticated and decrypted before load.**
- **OP.STORE of an object to OB.AROT_STORAGE is allowed if the following conditions hold:**
  - **The operation is performed by S.API.**
  - **The store request comes from an instance of the owner of the trusted storage space (S.API.caller = OB.AROT_STORAGE.owner).**
  - **OB.AROT_STORAGE is bound to the HUK OB.HUK (OB.AROT_STORAGE.ROT_identity = OB.HUK.ROT_identity).**
  - **If OB.AROT_STORAGE is located in external memory accessible to the NSPE (OB.AROT_STORAGE.inExtMem = True), then the object is signed and encrypted before storage.**

FDP_ACF.1.3/Trusted Storage The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly authorise access of subjects to objects]**.

FDP_ACF.1.4/Trusted Storage The TSF shall explicitly deny access of subjects to objects based on the following additional rules:

- **Any access to a trusted storage attempted from S.API without valid caller (S.API.caller = undefined)**
- **Any access to a trusted storage that was bound to a different MCU RoT (OB.AROT_STORAGE.ROT_identity different from OB.HUK.ROT_identity)**
- **Any access to a trusted storage from a subject different from S.API**
- **[assignment: rules, based on security attributes, that explicitly deny access of subjects to objects].**

7.1.1.8.3 FDP_ITT.1/Trusted Storage Basic internal transfer protection

FDP_ITT.1.1/Trusted Storage The TSF shall enforce the **Trusted Storage Access Control SFP** to prevent the **disclosure and modification** of user data when it is transmitted between physically-separated parts of the TOE.

7.1.1.8.4 FDP_ROL.1/Trusted Storage Basic rollback

FDP_ROL.1.1/Trusted Storage The TSF shall enforce **Trusted Storage Access Control SFP** to permit the rollback of the **unsuccessful or interrupted OP.STORE operation** on the **storage**.

FDP_ROL.1.2/Trusted Storage The TSF shall permit operations to be rolled back within the **[assignment: boundary limit to which rollback may be performed]**.

Application Note:

This SFR enforces atomicity of any write operation.

7.1.1.8.5 FMT_MSA.1/Trusted Storage Management of security attributes

FMT_MSA.1.1/Trusted Storage The TSF shall enforce the **Trusted Storage Access Control SFP** to restrict the ability to **query** the security attributes **OB.AROT_STORAGE.owner**, **OB.AROT_STORAGE.inExtMem**, **OB.AROT_STORAGE.ROT_identity**, and **OB.HUK.ROT_identity** to **ARoT_User** role.

7.1.1.8.6 FMT_MSA.3/Trusted Storage Static attribute initialisation

FMT_MSA.3.1/Trusted Storage The TSF shall enforce the **Trusted Storage Access Control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/Trusted Storage The TSF shall allow the **ARoT_User** to specify alternative initial values to override the default values when an object or information is created.

7.1.1.9 Attestation

7.1.1.9.1 FCO_NRO.1/Attestation Selective proof of origin

FCO_NRO.1.1/Attestation The TSF shall be able to generate evidence of origin for transmitted **attestation reports** at the request of the **recipient**.

FCO_NRO.1.2/Attestation The TSF shall be able to relate the **attestation key and the MCU RoT Identifier** of the originator of the information, and the **attestation tokens part of the attestation report** of the information to which the evidence applies.

FCO_NRO.1.3/Attestation The TSF shall provide a capability to verify the evidence of origin of information to **the recipient** given **the attestation key used in attestation report signature**.

7.1.2 MCU RoT Persistent Time PP-Module

The following security functional components defined in CC Part 2 [CC2] are used in this PP-Module:

- FMT_MTD.1 Management of TSF data
- FMT_SMF.1 Specification of Management Functions
- FPT_STM.1 Reliable time stamps

7.1.2.1.1 FMT_MTD.1/Persistent Time Management of TSF data

FMT_MTD.1.1/Persistent Time The TSF shall restrict the ability to **perform a 'time setting' operation on** the **ARoT persistent time** to **any instance of the ARoT**.

Application Note:

The 'time setting' operation can only affect the persistent time value of the ARoT performing the operation.

7.1.2.1.2 FMT_SMF.1/Persistent Time Specification of Management Functions

FMT_SMF.1.1/Persistent Time The TSF shall be capable of performing the following management functions: **'time setting' operation for ARoT persistent time**.

Application Note:

The 'time setting' operation can only affect the persistent time value of the ARoT performing the operation.

7.1.2.1.3 FPT_STM.1/Persistent Time Reliable time stamps

FPT_STM.1.1/Persistent Time The TSF shall be able to provide reliable time stamps.

Refinement:

The TSF shall be able to provide time stamps to ARoT instances such that:

- Time stamps are persistent over MCU RoT reset.
- Time stamps are monotonic between two 'time setting' operations performed by any instance of the ARoT.

The TSF shall invalidate any persistent time that does not meet the monotonicity property.

Application Note:

The refinement provides the meaning of the reliability that is expected.

7.1.3 MCU RoT Debug PP-Module

The following security functional components defined in CC Part 2 [CC2] are used in this PP-Module:

- FCS_COP.1 Cryptographic operation
- FDP_ACC.1 Subset access control
- FDP_ACF.1 Security attribute based access control
- FIA_ATD.1 User attribute definition
- FIA_UAU.2 User authentication before any action
- FIA_UAU.6 Re-authenticating
- FIA_UID.2 User identification before any action
- FIA_USB.1 User-subject binding
- FMT_SMR.1 Security roles

The additional *user* introduced by this PP-Module is:

- MCU RoT Debug Administrator

The additional *subject* introduced by this PP-Module is:

- S.DEBUG: The debug interface, with security attributes "enabled" (True/False) to state whether this feature is available on the TOE (attribute set before TOE delivery and not modifiable afterwards) and "authenticated" (True/False) to state whether the MCU RoT Debug Administrator has been authenticated.

This PP-Module allows *debug operations* performed by S.DEBUG on behalf of the MCU RoT Debug Administrator:

- OP.AUTHENTICATE: Activation of the debug feature by MCU RoT Debug Administrator authentication
- OP.DEBUG: Debug operations.

This PP-Module defines the following access control and information flow security functional policies (SFP):

Debug access control SFP:

- Purpose: To control the access to debug facilities of the MCU RoT.
- Subjects: S.DEBUG
- Objects: All
- Security attributes: S.DEBUG.enabled, S.DEBUG.authenticated
- Operations: OP.AUTHENTICATE, OP.DEBUG
- SFR instances: FDP_ACC.1/Debug, FDP_ACF.1/Debug.

7.1.3.1.1 FCS_COP.1/Debug Cryptographic operation

FCS_COP.1.1/Debug The TSF shall perform **authentication of the MCU RoT Debug Administrator or the actor acting on his behalf** in accordance with a specified cryptographic algorithm **[assignment: cryptographic algorithm]** and cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignment: list of standards]**.

7.1.3.1.2 FDP_ACC.1/Debug Subset access control

FDP_ACC.1.1/Debug The TSF shall enforce the **Debug access control SFP** on:

- **Subjects: S.DEBUG**
- **Objects: all objects**
- **Operations: OP.AUTHENTICATE, OP.DEBUG.**

7.1.3.1.3 FDP_ACF.1/Debug Security attribute based access control

FDP_ACF.1.1/Debug The TSF shall enforce the **Debug access control SFP** to objects based on the following:

- **S.DEBUG.enabled, S.DEBUG.authenticated**
- **[assignment: list of subjects and objects controlled under the indicated SFP, and for each, the SFP-relevant security attributes, or named groups of SFP-relevant security attributes].**

FDP_ACF.1.2/Debug The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **OP.AUTHENTICATE is allowed if the following conditions hold:**
  - **The operation is performed by S.DEBUG.**
  - **The debug interface is enabled (S.DEBUG.enabled = True).**
- **OP.DEBUG on all objects is allowed if the following conditions hold:**
  - **The operation is performed by S.DEBUG.**
  - **The debug interface is enabled (S.DEBUG.enabled = True).**
  - **The MCU RoT Debug Administrator is authenticated (S.DEBUG.authenticated = True).**
- **[assignment: rules governing access among controlled subjects and controlled objects using controlled operations on controlled objects].**

FDP_ACF.1.3/Debug The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly authorise access of subjects to objects]**.

FDP_ACF.1.4/Debug The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly deny access of subjects to objects]**.

7.1.3.1.4 FIA_ATD.1/Debug User attribute definition

FIA_ATD.1.1/Debug The TSF shall maintain the following list of security attributes belonging to individual users: **S.DEBUG.enabled**, **S.DEBUG.authenticated**.

7.1.3.1.5 FIA_UID.2/Debug User identification before any action

FIA_UID.2.1/Debug [Editorially Refined] The TSF shall require each user to be successfully identified before allowing any other TSF-mediated **debug** actions on behalf of that user.

7.1.3.1.6 FIA_USB.1/Debug User-subject binding

FIA_USB.1.1/Debug The TSF shall associate the following user security attributes with subjects acting on the behalf of that user: **S.DEBUG.enabled**, **S.DEBUG.authenticated**.

FIA_USB.1.2/Debug The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **S.DEBUG.authenticated is False**.

FIA_USB.1.3/Debug The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users:

- **S.DEBUG.authenticated is set to True after MCU RoT Debug Administrator successful authentication.**
- **S.DEBUG.authenticated is set to False when the authentication is lost, for instance after power-off (cf. rules of FIA_UAU.6).**
- **[assignment: rules for the changing of attributes].**

7.1.3.1.7 FMT_SMR.1/Debug Security roles

FMT_SMR.1.1/Debug The TSF shall maintain the roles "**MCU RoT Debug Administrator**".

FMT_SMR.1.2/Debug The TSF shall be able to associate users with roles.

Application Note:

The MCU RoT Debug Administrator is not intended to be the end-user, but someone involved in the life cycle of the product and who has access to the debug credential set during phase 3 or 5.

7.1.4 MCU RoT ARoT Isolation PP-Module

The following security functional component defined in CC Part 2 [CC2] is used in this PP-Module:

- FDP_IFF.1 Simple security attributes

The SFR FDP_IFF.1.1/Runtime_ARoT defined in this PP-Module refines and replaces the SFR FDP_IFF.1.1/Runtime defined in the core MCU RoT PP. It provides isolation between Application Roots of Trust within SPE. The first rule of FDP_IFF.1.1/Runtime is modified in order to consider access rights to the S.RAM_UNIT for each individual S.AROT_INSTANCE.

7.1.4.1.1 FDP_IFF.1/Runtime_ARoT Simple security attributes

FDP_IFF.1.1/Runtime_ARoT The TSF shall enforce the **Runtime Data Information Flow Control SFP** based on the following types of subject and information security attributes: **S.RESOURCE.state, S.RAM_UNIT.rights, and S.API.caller**.

FDP_IFF.1.2/Runtime_ARoT The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

- **Rules for information flow between S.AROT_INSTANCE and S.RAM_UNIT:**
  - **Flow of I.RUNTIME_DATA from S.AROT_INSTANCE to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(S.AROT_INSTANCE) is Write or ReadWrite.**
  - **Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.AROT_INSTANCE is allowed only if S.RAM_UNIT.rights(S.AROT_INSTANCE) is Read or ReadWrite.**
- **Rules for information flow from and to S.COMM_AGENT:**
  - **Flow of I.RUNTIME_DATA from S.COMM_AGENT to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(NSPE) is Write or ReadWrite.**
  - **Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.COMM_AGENT is allowed only if S.RAM_UNIT.rights(NSPE) is Read or ReadWrite.**
- **Rules for information flow from and to S.API:**
  - **Flow of I.RUNTIME_DATA from S.API to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(S.API.caller) is Write or ReadWrite.**
  - **Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.API is allowed only if S.RAM_UNIT.rights(S.API.caller) is Read or ReadWrite.**
- **Rules for information flow from and to S.RESOURCE:**
  - **Flow of I.RUNTIME_DATA between S.API and S.RESOURCE is allowed only if the resource is under MCU RoT control (S.RESOURCE.state = RoT).**

FDP_IFF.1.3/Runtime_ARoT The TSF shall enforce the **[assignment: additional information flow control SFP rules]**.

FDP_IFF.1.4/Runtime_ARoT The TSF shall explicitly authorise an information flow based on the following rules:

- **Rules for information flow from and to S.AROT_INSTANCE_SESSION:**
  - **Flow of I.RUNTIME_DATA that are parameter or return values is allowed between S.AROT_INSTANCE_SESSION and S.COMM_AGENT.**
  - **Flow of I.RUNTIME_DATA that are parameter or return values is allowed between S.AROT_INSTANCE_SESSION and S.API.**

FDP_IFF.1.5/Runtime The TSF shall explicitly deny an information flow based on the following rules: **Any information flow involving an MCU RoT subject unless one of the conditions stated in FDP_IFF.1.1/1.2/1.3/1.4 holds.**

7.2 Security Assurance Requirements

The Protection Profile provides a set of Security Assurance Requirements (SARs) which consists of the EAL 2 predefined package augmented with ALC_FLR.2 and with AVA_VAN_AP.3. This SAR raises the AVA_VAN.2 Basic attack potential to an Enhanced-basic attack potential.

As both AVA_VAN.2 and AVA_VAN_AP.3 are selected in the augmented EAL, the evaluator should perform two attack quotations according to the quotation grids associated with each of these SARs.

7.3 Security Requirements Rationale

7.3.1 Security Objectives for the TOE

7.3.1.1 Core MCU RoT PP

O.AROT_AUTHENTICITY The following requirements contribute to fulfil the objective:

- FCS_COP.1/Internals states the cryptography used to verify the authenticity of ARoT code.
- FDP_SDI.2 enforces the integrity and authenticity of ARoT code during storage.
- FPT_TEE.1 enforces the check of authenticity of ARoT code prior execution.

O.ATTESTATION The following requirements contribute to fulfil the objective:

- FCS_COP.1/Internals states the cryptography used to perform measurements of software components and to sign attestation reports.
- FCO_NRO.1/Attestation enforces generation of proof of origin for the attestation report.

O.CA_AROT_IDENTIFICATION The following requirements contribute to fulfil the objective:

- FIA_ATD.1 enforces the management of the Client and ARoT identity and properties as security attributes, which then become TSF data, protected in integrity and confidentiality.

- FIA_UID.2 requires the identification of Client application or ARoT before any action, thus allowing access to services and data to authorised users only.
- FIA_USB.1 enforces the association of the user identity to the active entity that acts on behalf of the user and ensures that identities are not modified

O.INITIALISATION The following requirements contribute to fulfil the objective:

- FCS_COP.1/Internals states the cryptography used to verify the authenticity of MCU RoT firmware.
- FPT_FLS.1 states that the MCU RoT has to reach a secure state upon initialisation or device binding failure.
- FPT_INI.1 enforces the initialisation of the TSF through a secure process including the verification of the authenticity and integrity of the MCU RoT firmware.

O.INSTANCE_TIME The following requirement fulfils the objective:

- FPT_STM.1/Instance time enforces the reliability of ARoT instance time.

O.KEYS_USAGE The following requirements contribute to fulfil the objective:

- FCS_COP.1/API and FCS_COP.1/Internals allow to specify the cryptographic operations in the scope of the evaluation.
- FDP_ACC.1/ARoT_keys, FDP_ACF.1/ARoT_keys, FMT_MSA.1/ARoT_keys, FMT_MSA.3/ARoT_keys, FMT_SMF.1 and FMT_SMR.1 state the key access policy, which grants access to the owner of the key only.

O.MCU_ROT_DATA_PROTECTION The following requirements contribute to fulfil the objective:

- FCS_COP.1/Internals states the cryptography used to protect integrity and confidentiality of the MCU RoT data in external memory, if applicable.
- FDP_SDI.2 monitors the authenticity and integrity of MCU RoT persistent data and states the response upon failure.
- FPT_ITT.1/Runtime enforces secure transmission and storage of MCU RoT persistent data.

O.MCU_ROT_ID The following requirements contribute to fulfil the objective:

- FAU_SAR.1 enforces MCU RoT identifier access capabilities.
- FAU_STG.1 enforces MCU RoT identifier storage capabilities.
- FCS_RNG.1 enforces uniqueness of the MCU RoT identifier if it is generated on the TOE.
- FPT_INI.1 enforces the integrity of the MCU RoT identifier, and it states the behaviour in case of failure.

O.MCU_ROT_ISOLATION The following requirements contribute to fulfil the objective:

- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state the policy for controlling access to the MCU RoT execution space.

O.OPERATION The following requirements contribute to fulfil the objective:

- FAU_ARP.1 states the MCU RoT responses to potential security violations.
- FDP_ACC.1/ARoT_keys, FDP_ACF.1/ARoT_keys, FMT_MSA.1/ARoT_keys, FMT_MSA.3/ARoT_keys, and FMT_SMF.1 state the key access policy.
- FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, and FMT_SMF.1 state the policy for controlling access to ARoT storage.
- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state the policy for controlling access to ARoT and MCU RoT execution spaces.
- FDP_SDI.2 enforces the monitoring of integrity and authenticity of MCU RoT data and ARoT, and it states the behaviour in case of failure.
- FIA_ATD.1, FIA_UID.2, and FIA_USB.1 ensure that actions are performed by identified users.
- FMT_SMR.1 states the two operational roles enforced by the MCU RoT.
- FPT_FLS.1 states that abnormal operations have to lead to a secure state.

Rationale specific to the MCU RoT Debug PP-Module:

- FDP_ACC.1/Debug, FDP_ACF.1/Debug, FIA_ATD.1/Debug, FIA_UAU.2/Debug, FIA_UAU.6/Debug, FIA_UID.2/Debug, FIA_USB.1/Debug and FMT_SMR.1/Debug state the debug access policy, which grants access to the debug facilities of the MCU RoT if this feature is not disabled.

O.RNG The requirement FCS_RNG.1 directly fulfils the objective.

O.ROLLBACK_PROTECTION The following requirements contribute to fulfil the objective:

- FDP_SDI.2 states the behaviour of the MCU RoT upon integrity failure (thus rollback)
- FPT_FLS.1 enforces the detection of integrity failure (thus rollback detection).

O.RUNTIME_CONFIDENTIALITY The following requirements contribute to fulfil the objective:

- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime ensure read access to authorised entities only.
- FDP_ITT.1/Runtime and FPT_ITT.1/Runtime ensure protection against disclosure of MCU RoT and ARoT data that is transferred between resources.
- FDP_RIP.1/Runtime states resource clean up policy.

O.RUNTIME_INTEGRITY The following requirements contribute to fulfil the objective:

- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state MCU RoT and ARoT runtime data policy, which grants write access to authorised entities only.
- FDP_ITT.1/Runtime and FPT_ITT.1/Runtime ensure protection against modification of MCU RoT and ARoT data that is transferred between resources.
- FDP_SDI.2 monitors the authenticity and integrity of RoT code, the RoT runtime data, the ARoT code and the ARoT data and keys and states the response upon failure.

O.TRUSTED_STORAGE The following requirements contribute to fulfil the objective:

- FCS_COP.1/Internals states the cryptography used to protect integrity and confidentiality of the ARoT data in external memory.
- FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FMT_SMF.1 and FMT_SMR.1 state the policy for accessing ARoT trusted storage and protecting the confidentiality of data.
- FDP_ITT.1/Trusted Storage ensures protection against disclosure of MCU RoT and ARoT data that is transferred between resources.
- FDP_SDI.2 enforces the integrity and authenticity of the trusted storage.
- FPT_FLS.1 maintains a secure state.
- FPT_INI.1 enforces the integrity of MCU RoT identifier and HUK, and it states the behaviour in case of failure.

7.3.1.2 MCU RoT Persistent Time PP-Module

O.AROT_PERSISTENT_TIME The following requirements fulfil the objective:

- FMT_MTD.1/Persistent Time states the roles that can perform 'time-setting' operations.
- FMT_SMF.1/Persistent Time states the existence of a 'time-setting' management function.
- FPT_STM.1/Persistent Time states the persistent time reliability conditions expected from the MCU RoT.

7.3.1.3 MCU RoT Debug PP-Module

O.DEBUG The following requirements contribute to fulfil the objective:

- FCS_COP.1/Debug allows to specify the cryptographic operations used for authenticating the MCU RoT Debug Administrator.
- FDP_ACC.1/Debug, FDP_ACF.1/Debug, FIA_ATD.1/Debug, FIA_UAU.2/Debug, FIA_UAU.6/Debug, FIA_UID.2/Debug, FIA_USB.1/Debug and FMT_SMR.1/Debug state the debug access policy, which grants access to the MCU RoT Debug Administrator only.

7.3.1.4 MCU RoT ARoT Isolation PP-Module

O.AROT_ISOLATION The following requirements contribute to fulfil the objective:

- FCS_COP.1/Internals states the cryptography used to protect integrity and confidentiality of the ARoT data.
- FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FMT_SMF.1 and FMT_SMR.1 state the policy for accessing ARoT trusted storage and protecting the confidentiality of data.
- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime_ARoT ensure read or write access to authorised entities only.
- FPT_FLS.1 maintains a secure state.

7.3.2 Mapping between TOE Security Objectives and SFRs

Table 14 and Table 15 present the mapping between security objectives for the TOE and SFRs.

Table 14: TOE Security Objectives and SFRs – Coverage

Core PP / PP-Modules	TOE Security Objectives	Security Functional Requirements	Rationale
Core PP	O.AROT_AUTHENTICITY	FCS_COP.1/Internals FDP_SDI.2 FPT_TEE.1	section 7.3.1
Core PP	O.ATTESTATION	FCS_COP.1/Internals FCO_NRO.1/Attestation	section 7.3.1
Core PP	O.CA_AROT_IDENTIFICATION	FIA_ATD.1 FIA_UID.2 FIA_USB.1	section 7.3.1
Core PP	O.INITIALISATION	FCS_COP.1/Internals FPT_FLS.1 FPT_INI.1	section 7.3.1
Core PP	O.INSTANCE_TIME	FPT_STM.1/Instance time	section 7.3.1
Core PP	O.KEYS_USAGE	FCS_COP.1/API FCS_COP.1/Internals FDP_ACC.1/ARoT_keys FDP_ACF.1/ARoT_keys FMT_MSA.1/ARoT_keys FMT_MSA.3/ARoT_keys FMT_SMF.1 FMT_SMR.1	section 7.3.1
Core PP	O.MCU_ROT_DATA_PROTECTION	FCS_COP.1/Internals FDP_SDI.2 FPT_ITT.1/Runtime	section 7.3.1
Core PP	O.MCU_ROT_ID	FAU_SAR.1 FAU_STG.1 FCS_RNG.1 FPT_INI.1	section 7.3.1
Core PP	O.MCU_ROT_ISOLATION	FDP_IFC.2/Runtime FDP_IFF.1/Runtime	section 7.3.1
Core PP	O.OPERATION	FAU_ARP.1 FDP_ACC.1/ARoT_keys FDP_ACF.1/ARoT_keys	section 7.3.1

Core PP / PP-Modules	TOE Security Objectives	Security Functional Requirements	Rationale
		FDP_ACC.1/Trusted Storage FDP_ACF.1/Trusted Storage FDP_IFC.2/Runtime FDP_IFF.1/Runtime FDP_SDI.2 FIA_ATD.1 FIA_UID.2 FIA_USB.1 FMT_MSA.1/ARoT_keys FMT_MSA.3/ARoT_keys FMT_MSA.1/Trusted Storage FMT_MSA.3/Trusted Storage FMT_SMR.1 FMT_SMF.1 FPT_FLS.1 and in the framework of Debug PP-Module also: FDP_ACC.1/Debug FDP_ACF.1/Debug FIA_ATD.1/Debug FIA_UAU.2/Debug FIA_UAU.6/Debug FIA_UID.2/Debug FIA_USB.1/Debug FMT_SMR.1/Debug	
Core PP	O.RNG	FCS_RNG.1	section 7.3.1
Core PP	O.ROLLBACK_PROTECTION	FDP_SDI.2 FPT_FLS.1	section 7.3.1
Core PP	O.RUNTIME_CONFIDENTIALITY	FDP_IFC.2/Runtime FDP_IFF.1/Runtime FDP_ITT.1/Runtime FDP_RIP.1/Runtime FPT_ITT.1/Runtime	section 7.3.1
Core PP	O.RUNTIME_INTEGRITY	FDP_IFC.2/Runtime FDP_IFF.1/Runtime FDP_ITT.1/Runtime FDP_SDI.2 FPT_ITT.1/Runtime	section 7.3.1
Core PP	O.TRUSTED_STORAGE	FCS_COP.1/Internals FDP_ACC.1/Trusted Storage FDP_ACF.1/Trusted Storage	section 7.3.1

Core PP / PP-Modules	TOE Security Objectives	Security Functional Requirements	Rationale
		FDP_ITT.1/Trusted Storage FDP_ROL.1/Trusted Storage FDP_SDI.2 FMT_MSA.1/Trusted Storage FMT_MSA.3/Trusted Storage FMT_SMF.1 FMT_SMR.1 FPT_FLS.1 FPT_INI.1	
ARoT Isolation PP-Module	O.AROT_ISOLATION	FCS_COP.1/Internals FDP_ACC.1/Trusted Storage FDP_ACF.1/Trusted Storage FDP_IFC.2/Runtime FDP_IFF.1/Runtime_ARoT FMT_MSA.1/Trusted Storage FMT_MSA.3/Trusted Storage FMT_SMF.1 FMT_SMR.1 FPT_FLS.1	section 7.3.1
Persistent time PP-Module	O.AROT_PERSISTENT_TIME	FMT_MTD.1/Persistent Time FMT_SMF.1/Persistent Time FPT_STM.1/Persistent Time	section 7.3.1
Debug PP-Module	O.DEBUG	FCS_COP.1/Debug FDP_ACC.1/Debug FDP_ACF.1/Debug FIA_ATD.1/Debug FIA_UAU.2/Debug FIA_UAU.6/Debug FIA_UID.2/Debug FIA_USB.1/Debug FMT_SMR.1/Debug	section 7.3.1

Table 15: SFRs and TOE Security Objectives - coverage

Core PP / PP-Modules	Security Functional Requirements	TOE Security Objectives
Core PP	FAU_ARP.1	O.OPERATION
Core PP	FAU_SAR.1	O.MCU_ROT_ID
Core PP	FAU_STG.1	O.MCU_ROT_ID
Core PP	FCO_NRO.1/Attestation	O.ATTESTATION

Core PP / PP-Modules	Security Functional Requirements	TOE Security Objectives
Core PP	FCS_COP.1/API[2]	O.KEYS_USAGE
Debug PP-Module	FCS_COP.1/Debug	O.DEBUG
Core PP ARoT Isolation PP-Module	FCS_COP.1/Internals	O.AROT_AUTHENTICITY O.AROT_ISOLATION (ARoT Isolation PP-Module) O.ATTESTATION O.INITIALISATION O.KEYS_USAGE O.MCU_ROT_DATA_PROTECTION O.TRUSTED_STORAGE
Core PP	FCS_RNG.1	O.MCU_ROT_ID O.RNG
Core PP	FDP_ACC.1/ARoT_keys	O.KEYS_USAGE O.OPERATION
Debug PP-Module	FDP_ACC.1/Debug	O.DEBUG O.OPERATION
Core PP ARoT Isolation PP-Module	FDP_ACC.1/Trusted Storage	O.AROT_ISOLATION (ARoT Isolation PP-Module) O.OPERATION O.TRUSTED_STORAGE
Core PP	FDP_ACF.1/ARoT_keys	O.KEYS_USAGE O.OPERATION
Debug PP-Module	FDP_ACF.1/Debug	O.DEBUG O.OPERATION
Core PP ARoT Isolation PP-Module	FDP_ACF.1/Trusted Storage	O.AROT_ISOLATION ARoT Isolation PP-Module) O.OPERATION O.TRUSTED_STORAGE
Core PP ARoT Isolation PP-Module	FDP_IFC.2/Runtime	O.AROT_ISOLATION (ARoT Isolation PP-Module) O.MCU_ROT_ISOLATION O.OPERATION O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY

[2] The OSP.CRYPTO_API is associated to this SFR.

Core PP / PP-Modules	Security Functional Requirements	TOE Security Objectives
Core PP ARoT Isolation PP-Module	FDP_IFF.1/Runtime	O.AROT_ISOLATION (ARoT Isolation PP-Module) O.MCU_ROT_ISOLATION O.OPERATION O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY
ARoT Isolation PP-Module	FDP_IFF.1/Runtime_ARoT	O.AROT_ISOLATION
Core PP	FDP_ITT.1/Runtime	O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY
Core PP	FDP_ITT.1/Trusted Storage	O.TRUSTED_STORAGE
Core PP	FDP_RIP.1/Runtime	O.RUNTIME_CONFIDENTIALITY
Core PP	FDP_ROL.1/Trusted Storage	O.TRUSTED_STORAGE
Core PP	FDP_SDI.2	O.AROT_AUTHENTICITY O.MCU_ROT_DATA_PROTECTION O.OPERATION O.ROLLBACK_PROTECTION O.RUNTIME_INTEGRITY O.TRUSTED_STORAGE
Core PP	FIA_ATD.1	O.CA_AROT_IDENTIFICATION O.OPERATION
Debug PP-Module	FIA_ATD.1/Debug	O.DEBUG O.OPERATION
Debug PP-Module	FIA_UAU.2/Debug	O.DEBUG O.OPERATION
Debug PP-Module	FIA_UAU.6/Debug	O.DEBUG O.OPERATION
Core PP	FIA_UID.2	O.CA_AROT_IDENTIFICATION O.OPERATION
Debug PP-Module	FIA_UID.2/Debug	O.DEBUG O.OPERATION
Core PP	FIA_USB.1	O.CA_AROT_IDENTIFICATION O.OPERATION
Debug PP-Module	FIA_USB.1/Debug	O.DEBUG O.OPERATION
Core PP	FMT_MSA.1/ARoT_keys	O.KEYS_USAGE O.OPERATION

Core PP / PP-Modules	Security Functional Requirements	TOE Security Objectives
Core PP	FMT_MSA.3/ARoT_keys	O.KEYS_USAGE O.OPERATION
Core PP ARoT Isolation PP-Module	FMT_MSA.1/Trusted Storage	O.AROT_ISOLATION (ARoT Isolation PP-Module) O.OPERATION O.TRUSTED_STORAGE
Core PP ARoT Isolation PP-Module	FMT_MSA.3/Trusted Storage	O.AROT_ISOLATION (ARoT Isolation PP-Module) O.OPERATION O.TRUSTED_STORAGE
Persistent Time PP-Module	FMT_MTD.1/Persistent Time	O.AROT_PERSISTENT_TIME
Core PP ARoT Isolation PP-Module	FMT_SMF.1	O.AROT_ISOLATION (ARoT Isolation PP-Module) O.KEYS_USAGE O.OPERATION O.TRUSTED_STORAGE
Persistent Time PP-Module	FMT_SMF.1/Persistent Time	O.AROT_PERSISTENT_TIME
Core PP	FMT_SMR.1	O.AROT_ISOLATION (ARoT Isolation PP-Module) O.KEYS_USAGE O.OPERATION O.TRUSTED_STORAGE
Debug PP-Module	FMT_SMR.1/Debug	O.DEBUG O.OPERATION
Core PP ARoT Isolation PP-Module	FPT_FLS.1	O.AROT_ISOLATION (ARoT Isolation PP-Module) O.INITIALISATION O.OPERATION O.ROLLBACK_PROTECTION O.TRUSTED_STORAGE
Core PP	FPT_INI.1	O.INITIALISATION O.MCU_ROT_ID O.TRUSTED_STORAGE
Core PP	FPT_ITT.1/Runtime	O.MCU_ROT_DATA_PROTECTION O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY

Core PP / PP-Modules	Security Functional Requirements	TOE Security Objectives
Core PP	FPT_TEE.1	O.AROT_AUTHENTICITY
Core PP	FPT_STM.1/Instance time	O.INSTANCE_TIME
Persistent Time PP-Module	FPT_STM.1/Persistent Time	O.AROT_PERSISTENT_TIME

7.3.3 Dependencies

7.3.3.1 SFRs Dependencies

Table 16 presents the CC dependencies of the SFRs that are included in the PP and indicates if they are satisfied and by which SFR.

Table 16: SFRs Dependencies

Core PP or PP-Module	Requirements	CC Dependencies	Satisfied Dependencies
Core PP	FAU_ARP.1	(FAU_SAA.1)	See 1)
Core PP	FAU_SAR.1	(FAU_GEN.1)	See 2)
Core PP	FAU_STG.1	(FAU_GEN.1)	See 3)
Core PP	FCO_NRO.1/Attestation	(FIA_UID.1)	FIA_UID.2
Core PP	FCS_COP.1/API	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	See 4) and 5)
Debug PP-Module	FCS_COP.1/Debug	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	See 6) and 7)
Core PP	FCS_COP.1/Internals	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	See 8) and 9)
Core PP	FCS_RNG.1	No Dependencies	
Core PP	FDP_ACC.1/ARoT_keys	(FDP_ACF.1)	FDP_ACF.1/ARoT_keys
Debug PP-Module	FDP_ACC.1/Debug	(FDP_ACF.1)	FDP_ACF.1/Debug
Core PP	FDP_ACC.1/Trusted Storage	(FDP_ACF.1)	FDP_ACF.1/Trusted Storage
Core PP	FDP_ACF.1/ARoT_keys	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/ARoT_keys, FMT_MSA.3/ARoT_keys

Core PP or PP-Module	Requirements	CC Dependencies	Satisfied Dependencies
Debug PP-Module	FDP_ACF.1/Debug	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/Debug See 10)
Core PP	FDP_ACF.1/Trusted Storage	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/Trusted Storage, FMT_MSA.3/Trusted Storage
Core PP	FDP_IFC.2/Runtime	(FDP_IFF.1)	FDP_IFF.1/Runtime
Core PP	FDP_IFF.1/Runtime	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/Runtime
ARoT Isolation PP-Module	FDP_IFF.1/Runtime_ARoT	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/Runtime See 11)
Core PP	FDP_ITT.1/Runtime	(FDP_ACC.1 or FDP_IFC.1)	FDP_IFC.2/Runtime
Core PP	FDP_ITT.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted Storage
Core PP	FDP_RIP.1/Runtime	No Dependencies	
Core PP	FDP_ROL.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted Storage
Core PP	FDP_SDI.2	No Dependencies	
Core PP	FIA_ATD.1	No Dependencies	
Debug PP-Module	FIA_ATD.1/Debug	No Dependencies	
Debug PP-Module	FIA_UAU.2/Debug	(FIA_UID.1)	FIA_UID.2/Debug
Debug PP-Module	FIA_UAU.6/Debug	No Dependencies	
Core PP	FIA_UID.2	No Dependencies	
Debug PP-Module	FIA_UID.2/Debug	No Dependencies	
Core PP	FIA_USB.1	(FIA_ATD.1)	FIA_ATD.1
Debug PP-Module	FIA_USB.1/Debug	(FIA_ATD.1)	FIA_ATD.1/Debug
Core PP	FMT_MSA.1/ARoT_keys	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.1/ARoT_keys FMT_SMF.1 FMT_SMR.1,

Core PP or PP-Module	Requirements	CC Dependencies	Satisfied Dependencies
Core PP	FMT_MSA.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.1/Trusted Storage FMT_SMF.1 FMT_SMR.1
Core PP	FMT_MSA.3/ARoT_keys	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/ARoT_keys FMT_SMR.1
Core PP	FMT_MSA.3/Trusted Storage	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/Trusted Storage FMT_SMR.1
Persistent Time PP-Module	FMT_MTD.1/Persistent Time	(FMT_SMF.1) and (FMT_SMR.1)	FMT_SMF.1/Persistent Time FMT_SMR.1
Core PP	FMT_SMF.1	No Dependencies	
Persistent Time PP-Module	FMT_SMF.1/Persistent Time	No Dependencies	
Core PP	FMT_SMR.1	(FIA_UID.1)	FIA_UID.2
Debug PP-Module	FMT_SMR.1/Debug	(FIA_UID.1)	FIA_UID.2/Debug
Core PP	FPT_FLS.1	No Dependencies	
Core PP	FPT_INI.1	No Dependencies	
Core PP	FPT_ITT.1/Runtime	No Dependencies	
Core PP	FPT_STM.1/Instance time	No Dependencies	
Persistent Time PP-Module	FPT_STM.1/Persistent Time	No Dependencies	
Core PP	FPT_TEE.1	No Dependencies	

The rationale for the exclusion of some of the SFRs' dependencies is the following:

1) **The dependency FAU_SAA.1 of FAU_ARP.1 is discarded.** The potential security violations are explicitly defined in the FAU_ARP.1 requirement. There is no audited event defined in the SFR of this PP.

2) **The dependency FAU_GEN.1 of FAU_SAR.1 is discarded.** This dependency is discarded as the only audit record considered is the MCU RoT identifier and this identifier is set before TOE delivery and non-modifiable afterwards.

3) **The dependency FAU_GEN.1 of FAU_STG.1 is discarded.** This dependency is discarded as the only audit record considered is the MCU RoT identifier and this identifier is set before TOE delivery and non-modifiable afterwards.

4) **The dependency FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2 of FCS_COP.1/API is discarded.** The component FCS_COP.1/API is present only for TOEs that provide cryptographic services to ARoTs. The ST author shall add the dependencies applicable to those services.

5) **The dependency FCS_CKM.4 of FCS_COP.1/API is discarded.** The component FCS_COP.1/API is present only for TOEs that provide cryptographic services to ARoTs. The ST author shall add the dependencies applicable to those services.

6) **The dependency FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2 of FCS_COP.1/Debug is discarded.** The MCU RoT Debug authentication key used for authenticating MCU RoT Debug Administrator in FCS_COP.1/Debug is set during manufacturing. It cannot be changed during the end-usage phase.

7) **The dependency FCS_CKM.4 of FCS_COP.1/Debug is discarded.** The MCU RoT Debug authentication key used for MCU RoT Debug Administrator authentication in FCS_COP.1/Debug is not required to be changed or destroyed during the end-usage phase.

8) **The dependency FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2 of FCS_COP.1/Internals is discarded.** The HUK cryptographic key used for cryptographic operations in FCS_COP.1/Internals is set during manufacturing. If a derived key is used for trusted storage, the ST author will have to add a dependency to FCS_CKM.1 and specify the derivation method.

9) **The dependency FCS_CKM.4 of FCS_COP.1/Internals is discarded.** The HUK used for cryptographic operations in FCS_COP.1/Internals is not required to be changed or destroyed during the end-usage phase.

10) **The dependency FMT_MSA.3 of FDP_ACF.1/Debug is discarded.** There is no management of security attributes by authorised users for this access control SFP as security attributes are either exclusively managed by the TSF or not modifiable during the end-usage phase, therefore the dependency FMT_MSA.3 is not applicable.

11) **The dependency FMT_MSA.3 of FDP_IFF.1/Runtime and FDP_IFF.1/Runtime_ARoT is discarded.** There is no management of security attributes by authorised users for this information flow control SFP as all security attributes are exclusively managed by the TSF, therefore the dependency FMT_MSA.3 is not applicable.

7.3.3.2 SARs Dependencies

Table 17 presents the CC dependencies of the SARs that are included in the PP and indicates if they are satisfied and by which SAR.

Table 17: SARs Dependencies

SAR	CC Dependencies	Satisfied Dependencies
ADV_ARC.1	(ADV_FSP.1) and (ADV_TDS.1)	ADV_FSP.2 ADV_TDS.1
ADV_FSP.2	(ADV_TDS.1)	ADV_TDS.1
ADV_TDS.1	(ADV_FSP.2)	ADV_FSP.2
AGD_OPE.1	(ADV_FSP.1)	ADV_FSP.2
AGD_PRE.1	No Dependencies	
ALC_CMC.2	ALC_CMS.1	ALC_CMS.2

SAR	CC Dependencies	Satisfied Dependencies
ALC_CMS.2	No Dependencies	
ALC_DEL.1	No Dependencies	
ALC_FLR.2	No Dependencies	
ASE_CCL.1	(ASE_ECD.1) and (ASE_INT.1) and (ASE_REQ.1)	ASE_ECD.1 ASE_INT.1 ASE_REQ.2
ASE_ECD.1	No Dependencies	
ASE_INT.1	No Dependencies	
ASE_OBJ.2	(ASE_SPD.1)	ASE_SPD.1
ASE_REQ.2	(ASE_ECD.1) and (ASE_OBJ.2)	ASE_ECD.1 ASE_OBJ.2
ASE_SPD.1	No Dependencies	
ASE_TSS.1	(ADV_FSP.1) and (ASE_INT.1) and (ASE_REQ.1)	ADV_FSP.2 ASE_INT.1 ASE_REQ.2
ATE_COV.1	(ADV_FSP.2) and (ATE_FUN.1)	ADV_FSP.2 ATE_FUN.1
ATE_FUN.1	(ATE_COV.1)	ATE_COV.1
ATE_IND.2	(ADV_FSP.2) and (AGD_OPE.1) and (AGD_PRE.1) and (ATE_COV.1) and (ATE_FUN.1)	ADV_FSP.2 AGD_OPE.1 AGD_PRE.1 ATE_COV.1 ATE_FUN.1
AVA_VAN.2	(ADV_ARC.1) and (ADV_FSP.2) and (ADV_TDS.1) and (AGD_OPE.1) and (AGD_PRE.1)	ADV_ARC.1 ADV_FSP.2 ADV_TDS.1 AGD_OPE.1 AGD_PRE.1
AVA_VAN_AP.3	(ADV_ARC.1) and (ADV_FSP.2) and (ADV_TDS.1) and (AGD_OPE.1) and (AGD_PRE.1)	ADV_ARC.1 ADV_FSP.2 ADV_TDS.1 AGD_OPE.1 AGD_PRE.1

7.3.4 Rationale for the Security Assurance Requirements

The assurance level defined in this Protection Profile consists of the predefined assurance package EAL 2 augmented with ALC_FLR_2 to mitigate exploitation of potential flaws discovered after TOE evaluation and with AVA_VAN_AP.3 in order to reach the Enhanced-basic attack potential.

This augmented EAL permits a developer to gain sufficient assurance from positive security engineering based on good commercial development practices that are compatible with industry constraints, in particular the life cycle of devices enabled with MCU RoTs. The developer must provide evidence of security engineering at design, testing, guidance, configuration management, and delivery levels as required by standard EAL 2. In order to cope with the high exposure of the MCU RoT and the interest that devices enabled with MCU RoTs and their embedded services may represent to attackers, the product has to show resistance to Enhanced-basic attack potential. This attack potential matches the threat analysis performed on typical architectures and attack profiles in the field.

The components AVA_VAN.2 and AVA_VAN_AP.3 are chosen together in the augmented EAL 2 package. The reason for this choice is to perform the attack quotation according to the two tables and to allow EAL 2 product recognition for the schemes that do not recognize the AVA_VAN_AP.3 component.