GLOBALPLATFORM®
THE STANDARD FOR SECURE DIGITAL SERVICES AND DEVICES

GlobalPlatform Technology
MCU Root of Trust Protection Profile
Version 0.2.0.0

Public Review

October 2020

Document Reference: GPT_SPE_146

Contents

Figures

Tables

1 Introduction

Title:	MCU Root of Trust Protection Profile (base-PP and optional RoT Persistent Time PP-module, RoT Debug PP-module and ARoT Isolation PP-module)
Identifications:	GPT_SPE_146 : PP-configuration composed of the base Protection Profile only
	GPT_SPE_146+PP-module(s) : PP-configuration composed of the base Protection Profile and a combination of the PP-modules defined in this document: RoT Persistent Time PP-module, RoT Debug PP-module and ARoT Isolation PP-module
Date:	October 2020
Version:	0.2.0.0
Sponsor:	GlobalPlatform
CC Version:	3.1 Revision 5

This Protection Profile (PP) has been developed by PSA JSA group then donated to GlobalPlatform Trusted Platform Services (TPS) Committee that has finalized the document. It constitutes the reference for the Common Criteria (CC) evaluation of microcontroller Root of Trust (MCU RoT), which aim at enabling IoT security services such as attestation, device authentication, personal data protection, etc.

The MCU Root of Trust in the scope of this PP implement an architecture neutral MCU Root of Trust. This PP relies on the Common Criteria Modular Protection Profile methodology [CC1] to define a ‘base-PP’ with the minimum RoT security requirements and optional ‘PP-modules’ that apply to those RoT that implement persistent monotonic time, Application Root of Trust isolation and those RoT that allow access to debug features. These ‘PP-modules’ can be used with the ‘base-PP’ to compose a ‘PP-configuration’. This document supports all the combinations of the ‘base-PP’ with the ‘PP-modules’ introduced above.

This Protection Profile claims conformance with the assurance package EAL 2+ which consists of the predefined EAL 2 augmented with ALC_FLR.2 and with AVA_VAN_AP.3. This extended SAR aims at raising the attack potential over the standard Basic attack potential defined for AVA_VAN.2 in [CC3] and [CEM]. The evaluation methodology, including the specific attack potential quotation table and a representative set of MCU attacks, is defined in a separate document. This extended SAR can only be used for product evaluation if it is recognized by the Certification Body that monitors the product evaluation otherwise the conformance claim is limited to EAL 2.

1.1 Audience

This document is dedicated to all actors in the MCU value chain: MCU developers, integrators, service providers (ARoT developers), as well as ITSEFs, certification bodies and Common Criteria certificate consumers.

1.2 IPR Disclaimer

GlobalPlatform draws attention to the fact that claims that compliance with this specification may involve the use of a patent or other intellectual property right (collectively, "IPR") concerning this specification may be published at https://www.globalplatform.org/specificationsipdisclaimers.asp. GlobalPlatform takes no position concerning the evidence, validity, and scope of these IPR claims.

1.3 References

Table 1-1: Normative References

Standard / Specification	Description	Ref.
BSI AIS 31	A proposal for: Functionality classes for random number generators. Version 2.0, 18 September 2011	[AIS31]
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]
CC Supporting Document	Application of Attack Potential to Smartcards. Version 3.0 April 2019. Joint Interpretation Library.	[APSC]
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]

Standard / Specification	Description	Ref.
NIST Special Publication 800-90B	Recommendation for the Entropy Sources Used for Random Bit Generation. January 2018	[SP800-90]
RFC2119	Key words for use in RFCs to Indicate Requirement Levels	[RFC2119]

1.4 Terminology and Definitions

Throughout this document, normative requirements are highlighted by use of capitalized key words as described below.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119]:

- MUST - This word, or the terms “REQUIRED” or “SHALL”, mean that the definition is an absolute requirement of the specification
- MUST NOT - This phrase, or the phrase “SHALL NOT”, mean that the definition is an absolute prohibition of the specification
- SHOULD - This word, or the adjective “RECOMMENDED”, mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course
- SHOULD NOT - This phrase, or the phrase “NOT RECOMMENDED” mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

MAY - This word, or the adjective “OPTIONAL”, mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

Table 1-2 defines the expressions used within this Protection Profile that use an upper case first letter in each word of the expression. Expressions within this document that use a lower case first letter in each word take the common sense meaning. CC terminology, defined in [CC1] §4, is not listed here.

Table 1-2: 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 RoT. Contrast *Client Application*.

Term	Definition
Client Application (CA)	An application running in the Non-Secure Processing Environment (NSPE) that accesses facilities provided by Application Root of Trust or Root of Trust Services inside the SPE. Contrast *Application Root of Trust*.
Device binding	Device binding is the property of data being only usable on a unique given system instance, here a RoT.
Execution Environment (EE)	A set of hardware and software components that provide facilities (computing, memory management, input/out, etc.) necessary to support applications.
Hardware Unique Key (HUK)	A persistent key, provisioned at device manufacture, used to bind client and device specific secrets to the RoT.
Monotonicity	Monotonicity is the property of a variable whose value is either always increasing or always decreasing over time.
Non-Secure Processing Environment (NSPE)	An environment that is provided and governed by a general purpose OS or RTOS, potentially in conjunction with other supporting operating systems and hypervisors; it is outside of the SPE. This environment and applications running on it are considered un-trusted. Contrast *Secure Processing Environment (SPE)*.
Power cycle	A power cycle is the lapse between the moment a device is turned on and the moment the device is turned off afterwards.
Production RoT	A RoT residing in a device that is in the end user phase of its life cycle.
NSPE Communication Agent	An NSPE OS driver that enables communication between the NSPE and the RoT.
NSPE OS	Typically, an OS providing a much wider variety of features than that of the OS running inside the SPE. It may be very open in its ability to accept applications. It will have been developed with functionality and performance as key goals, rather than security. Due to the size and needs of the NSPE OS it will run in an execution environment that may be larger than the SPE with much lower physical security boundaries. From the SPE viewpoint, everything in the NSPE has to be considered un-trusted, though from the NSPE OS point of view there may be internal trust structures. Contrast *NSPE OS*.
Root of Trust (RoT)	Generally, the smallest distinguishable set of hardware, firmware, and/or software that must be inherently trusted and which is closely tied to the logic and environment on which it performs its trusted actions.

Term	Definition
Secure Processing Environment (SPE)	An execution environment that runs alongside but isolated from an NSPE. A SPE has security capabilities and meets certain security-related requirements: It protects RoT assets from general software attacks, defines rigid safeguards as to data and functions that a program can access, and resists a set of defined threats. There are multiple technologies that can be used to implement a SPE, and the level of security achieved varies accordingly. Contrast *Non-Secure Processing Environment.*
Secure Partition Manager (SPM)	The operating system running in the SPE. It manages the isolation of 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.
System-on-Chip (SoC)	An electronic system all of whose components are included in a single integrated circuit.
ARoT instance time / ARoT persistent time	Time value available to an Application Root of Trust through API. The API offers two types of time values: System Time, which exists only during runtime, and Persistent time, which persists over resets. System Time must be monotonic for a given ARoT instance, and the returned value is called “ARoT instance time”. Persistent time depends only on the ARoT but not on a particular instance, it must be monotonic even across power cycles. Its monotonicity across power cycles is related to the Persistent Time optional PP-module.
NSPE Client API	The software interface used by clients running in the NSPE to communicate with the RoT and with the Applications Root of Trust executed on the SPE.
RoT Internal API	The software interface exposing RoT functionality to Applications Root of Trust.
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. A SPE Trusted Storage is not considered hardware tamper resistant to the levels achieved by Secure Elements.

1.5 Abbreviations and Notations

Table 1-3: 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])

Abbreviation	Meaning
CEM	Common Evaluation Methodology (defined in [CEM])
CM	Configuration Management (defined in [CC1])
EAL	Evaluation Assurance Level (defined in [CC1])
EE	Execution Environment
ID	IDentifier
HD	High-Definition
HUK	Hardware Unique Key
JTAG	Joint Test Action Group (defined in [JTAG])
MAC	Message Authentication Code
MCU	Microcontroller
N/A	Not Applicable
NSPE	Non-Secure Processing Environment
OMTP	Open Mobile Terminal Platform
OS	Operating System
OSP	Organizational Security Policy (defined in [CC1])
OTP	One-Time Password
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
RTOS	Real Time Operating System
RSA	Rivest / Shamir / Adleman asymmetric algorithm (defined in [RSA])
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])
SPE	Secure Processing Environment
SPM	Secure Partition Manager
SoC	System-on-Chip
SPD	Security Problem Definition (defined in [CC1])

Abbreviation	Meaning
SSL	Secure Sockets Layer
ST	Security Target (defined in [CC1])
TLS	Transport Layer Security
TOE	Target of Evaluation (defined in [CC1])
TSF	TOE Security Functionality (defined in [CC1])
TSFI	TSF Interface (defined in [CC1])
USB	Universal Serial Bus

1.6 Revision history

Date	Version	Description
20/06/2019	ALP01	Initial version based on GP TEE PP
18/12/2019	BET01	Candidate release for donation to GP
April 2020	0.1	Committee Review
August 2020	0.1	Member Review
October 2020	0.2	Public Review

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 usages as well as the TOE's life cycle.

2.1 TOE Type

The TOE type is the Root of Trust (RoT) for microcontrollers (MCUs). It is based on the security principles set out in the PSA Device Security Model (ARM DEN 0079) and potentially implementing the specifications PSA Firmware Framework and RoT Services – M-profile (ARM DEN 0063A). However, the PP does not require functional compliance with these standards.

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 hosts a set of Applications RoT (ARoT) and provides them with a comprehensive set of Root of Trust security services including: integrity of execution, trusted storage, key management and cryptographic algorithms, time management, attestation.

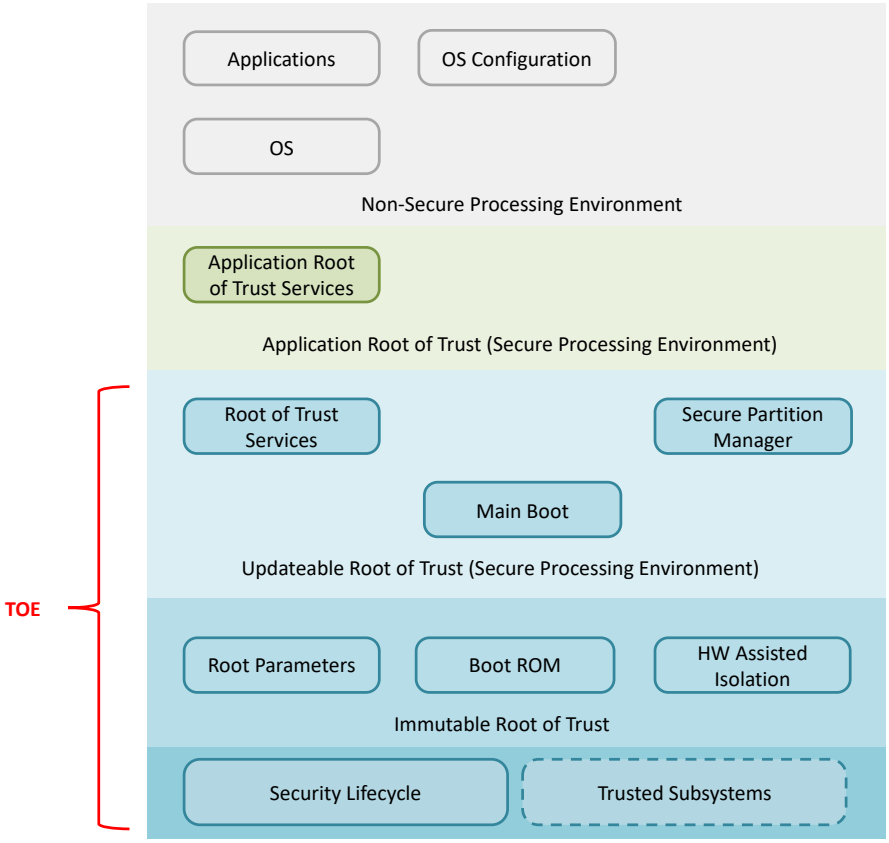
The TOE, as depicted in Figure 2-1, comprises:

- Any hardware, firmware and software used to provide the RoT security functionality. This includes:
  - Immutable root of trust, for example Boot ROM, Root parameters (such as initialization data, hardware keys and IDs), Isolation hardware, Security lifecycle management and enforcement (such as Debug feature if present). This component cannot be updated
  - Updateable root of trust, such as Software isolation framework, protecting more trusted software from less trusted software, Generic RoT services such as binding, initial attestation, generic crypto services, firmware update validation.
  - Trusted Subsystems used by the root of trust, such as security subsystems, trusted peripherals, SIM or SE, which include both hardware and software components, are also in the scope of evaluation.
- The guidance for the secure usage of the RoT after delivery.

The TOE does not comprise:

- The Application RoT
- The Non-Secure Processing Environment
- The NSPE Applications.

Figure 2-1: Scope of evaluation



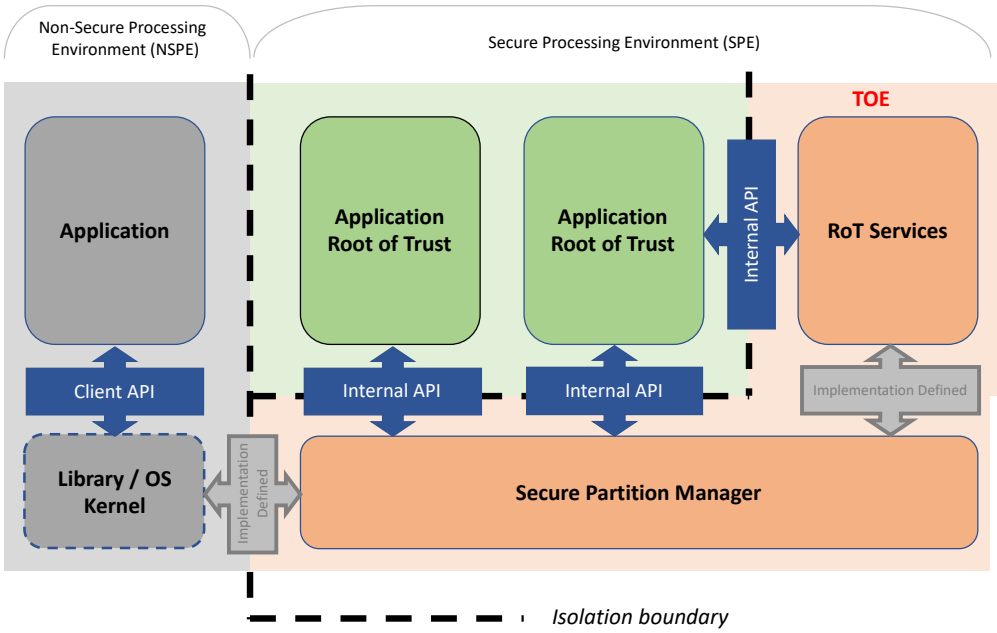
In the following, TOE and RoT are used interchangeably.

2.2 TOE Description

2.2.1 Software Architecture of a RoT-enabled MCU Device

The RoT 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 a RoT-enabled MCU device and the TOE, independently of any hardware architecture.

Figure 2-2: MCU RoT Overall Software Architecture



The RoT software architecture identifies two distinct classes of components:

- The Applications Root of Trust that run on the SPE and use the RoT Internal API
- The Root of Trust Services that provide security services to the SPE and NSPE

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

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

The TOE software external interface comprises the RoT Internal API (used by the Applications RoT) and the Client API (used by the NSPE).

The communication protocol between the NSPE and the SPE, used below the Client API level, is implementation-dependent, and therefore this Protection Profile does not mandate any particular such protocol. The security targets conformant to this PP shall describe all software interfaces used for communication with the NSPE and the SPE.

2.2.2 Hardware Architecture of a RoT-enabled MCU Device

The RoT is embedded in a device platform including:

- Microcontroller processing unit(s)
- Hardware resources:
  - Physical volatile memory
  - Physical non-volatile memory
  - Peripherals, like network devices
  - Cryptographic accelerators
  - Secure hardware clock

- A set of connections between the processing unit(s) and the hardware resources

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

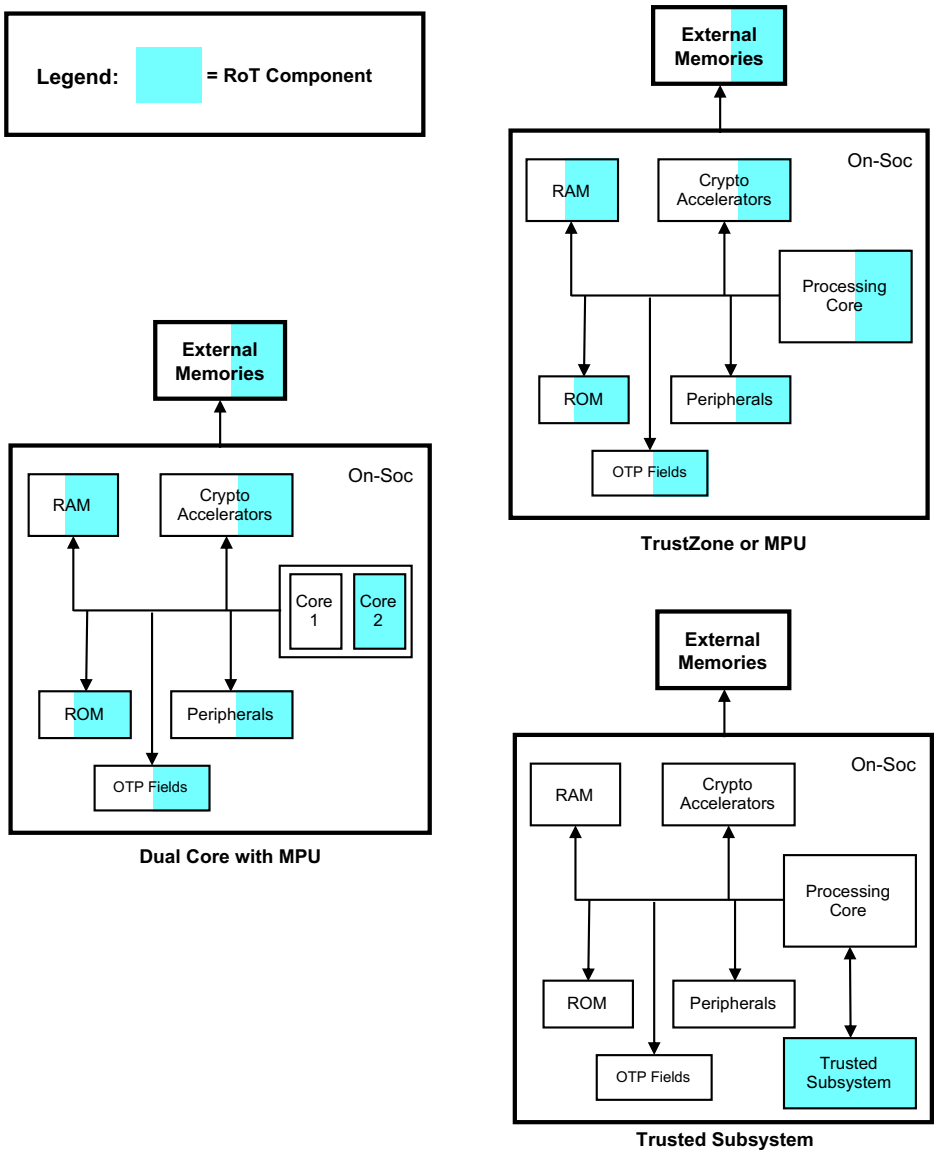
- The *die layer*, System-on-Chip (SoC), which contains processor(s) and resources such as memories, crypto-accelerators, peripherals (e.g. JTAG, USB, serial), etc.
- The *package layer*, which embeds the SoC and contains further resources, e.g. non-volatile and volatile memories, pins or buses. Resources 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 SoC, package, non-volatile and volatile memories, wireless and contactless interface chips and other resources.

The RoT is typically implemented in the die and package layers of one package but it may be instantiated in a number of separate packages using cryptographic linking (secure channels) between RoT components. The RoT hardware external interface stands for 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 a black-box.

Nevertheless, the physical boundary of the RoT is implementation-dependent. Furthermore, the set of “trusted” hardware resources used to realize the security functionality, which is controlled by the RoT, can change dynamically during execution of the RoT. For instance, some communication resources such as the network device may sometimes be within the RoT boundaries if the RoT enforces exclusive access to these resources. From a logical point of view, the “trusted” resources used by the RoT are separated from the “un-trusted” resources used by the NSPE. That is, the RoT in SPE and the NSPE coexist in the device but hardware-isolated from each other.

In practice, there are several ways to architect a RoT on an MCU device and to isolate it from the NSPE. Figure 2-3 illustrates three possible realizations, with different resource-sharing policies between the RoT on the SPE and the NSPE. Indeed, the SPE and the NSPE can share device resources provided the RoT controls access to them.

Figure 2-3: MCU RoT Overall Software Architecture



This Protection Profile does not mandate any particular hardware architecture, resource set or isolation mechanisms from the NSPE. The security targets conformant to this PP shall describe the physical layout and precisely define the physical boundaries of the RoT and the hardware external interface.

2.3 Usage and Major Security Features of the TOE

The purpose of the RoT is to provide secure service and to host and execute Applications RoT securely, enforcing their mutual isolation and isolation from other execution environments, and ensuring integrity and confidentially of the assets managed by the RoT.

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

2.3.1 RoT Security Functionality

The RoT security functionality in the end-user phase (cf. Section 2.5) which is in the scope of the evaluation consists of:

- Secure RoT instantiation through a secure initialization process using assets bound to the SoC, that ensures the authenticity and contributes to the integrity of the RoT code running in the device
- Isolation of the SPM, RoT services, the RoT resources involved and all the Applications Root of Trust from the NSPE
- Isolation of the SPM and RoT services from Applications Root of Trust
- Trusted storage of ARoT and RoT data and keys, ensuring integrity, confidentiality, atomicity and binding to the RoT
- Random Number Generator
- Cryptographic API including:
- Generation and derivation of keys and key pairs
- Support for cryptographic algorithms such as SHA-256, AES 128/256, TDES, RSA 2048, etc. (this list is for example only, see the Application Note below)
- 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 that particular combination of hardware and firmware of the TOE can be trusted
- Monotonic ARoT instance time
- Correct execution of SPM and RoT services
- RoT firmware integrity verification
- Prevention of downgrade of RoT firmware.

The RoT security functionality defines the logical boundary of the TOE. The interfaces of this boundary are 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 Applications RoT is out of the scope of the TOE.

Application Note:

Security Targets conformant to this PP 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 RoT enables the use of MCU-based IoT devices for a wide range of services that require security protection, for instance:

- Metering
- User authentication
- Personal data protection

- Infrastructure for IoT Edge.

2.3.3 RoT Persistent Time PP-Module

The RoT Persistent Time PP-module addresses the following security functionality, which complements the core functionality defined in section 2.3.1 with monotonic ARoT persistent time.

Notice 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 RoT Debug PP-Module

The RoT Debug PP-module addresses the authenticated access to RoT Debug functionality for the RoT Debug Administrator. In the base-PP this functionality is not supported and debug features are assumed to be deactivated prior delivery of the TOE to the end-user.

2.3.5 ARoT Isolation PP-Module

The ARoT Isolation PP-module extends the isolation properties already addressed in the base-PP to isolation between Applications Root of Trust.

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 behavior of non-TOE hardware, software or firmware.

Application note:

Security Targets conformant to this PP shall complete the descriptions of the available non-TOE hardware/software/firmware with the list of non-TOE resources required for the use of the TOE.

2.5 Reference TOE Life Cycle

The device life cycle outlined here is a reference life cycle from which implementations can deviate according to development, manufacturing and assembly processes. It is split in seven phases:

- Phase 1 corresponds to the design of firmware, software and hardware; it covers both MCU, RoT and additional components
- Phase 2 corresponds to the overall design of the hardware platform supporting the MCU
- Phase 3 corresponds to chipset and other hardware components manufacturing
- Phase 4 covers software preparation (e.g. linking the RoT software and other software)
- Phase 5 consists of TOE assembling; it includes any initialization and configuration step necessary to bring the TOE to a secure state prior 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 RoT initialization code, is usually installed in phase 3 though it may be upgraded later. The Hardware Unique Key (HUK) and the 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 RoT Services are usually installed after this step, in phase 3 or 5 though it may be upgraded later. Applications Root of Trust may be installed together with or after the SPM, either in step 3 or in step 5 – for this latter case, they may have been linked with the SPM in step 4. If the TOE supports RoT Debug functionalities, the flag to indicate whether the functionality is enabled on the RoT 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 phase 3 to phase 5, but must necessarily follow the setting of the HUK and of the RoT unique identifier, of the Debug enabled flag and the Debug credentials (if 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 are covered through the AGD assurance class by organizational security policies and security objectives for the environment
- The security of the end-usage environment is covered by the TOE security functionalities and by security objectives for the environment.

Table 2-1 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 steps represented, are possible. Cases where the SPE runs on a discrete separate processor, or where the RoT is installed by the chipset manufacturer in case 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 RoT integration, can result in different flows.

Table 2-1: Actors in the TOE Life Cycle

Phases	Actors
1 & 2: Firmware / Software / Hardware design	The RoT software developer Is in charge of RoT software development and testing May also develop the RoT initialization code that instantiates/initializes the RoT (e.g. part of the secure boot code) Specifies the RoT software linking requirements The device manufacturer may design additional NSPE software that will be linked with the RoT in phase 4 to provide NSPE-controlled resources. He may also design Applications Root of Trust that he will integrate in phase 4. The RoT hardware designer is in charge of designing (part of) the processor(s) where the RoT software runs and designing (part of) the hardware security resources used by the RoT. The chip vendor designs the ROM code and the secure portion of the RoT chipset. If the chip vendor is not designing the full RoT hardware, the chip vendor integrates (and potentially augments) the RoT hardware designed by the RoT hardware designer(s).

Phases	Actors
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 steps by the Chip Vendor and the purchasing OEM (step 4) through the programming of fuses (enables, sets or seeds the Hardware Unique Key).
4: Software integration	The OEM is responsible for the integration, validation and preparation of the software to load in the product that will include the RoT, any pre-installed Applications Root of Trust, and additional software required to use the product (e.g. NSPE, Client Applications).
5: Device manufacturing	The OEM is responsible for the device assembly, initialization, provisioning and any other operation on the TOE (including loading or installation of Applications Root of Trust) and device before delivery to the end user.
6: Deployed / Secure Enabled	The end user gets a device ready for use, including the TOE. This state only permits configuration operations that support the required use cases and has accesses to the security functions of the device. Its debugging and testing features are disabled and secure boot is mandatory. The TOE may be updated if it has not been designed to be immutable.
7: Return Material Authorization (RMA)	This is a terminal state used for devices that are returned to the 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:

Security Targets shall describe the actual TOE life cycle, identify the actors and development/manufacturing sites involved; they shall identify the actual integration points of the components (SPM and RoT services, HUK, ARoTs) into the TOE, as well as the actual delivery point of the TOE, and precise the process for setting the Hardware Unique Key and the phase in which it occurs.

Security targets shall also identify the TOE and the components that are delivered with the TOE if any, e.g. pre-installed Applications Root 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 Protection Profile, it must be described in the ST as well.

3 Conformance Claims and Consistency Rationale

This document uses modular PP methodology [CC1] and includes a base-PP and three PP-modules. This section applies to the base-PP and to the RoT Persistent Time, RoT Debug and ARoT Isolation PP-modules.

3.1 Conformance Claim to CC

This base Protection Profile is CC Part 2 [CC2] extended and CC Part 3 [CC3] extended. The CC Part 2 is extended with the security functional components FCS_RNG.1 Random numbers generation and FPT_INI.1 TSF initialization.

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

The RoT Persistent Time, RoT Debug and ARoT 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, defined in section 6.1.4.

This conformance claim also applies to the PP-configurations defined in this document.

3.3 Conformance Claim of the PP

This PP does not claim conformance to any another PP.

3.4 Conformance Claim to the PP

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

This conformance claim also applies to the PP-configurations defined in this document.

3.5 Consistency Rationale for the PP-Modules

3.5.1 RoT Persistent Time PP-Module

The RoT Persistent Time PP-Module is intended to be used only with the base-PP of this document. It complements the RoT functionalities defined in section 2.3.1 that form the base-PP. It defines a new functionality which is the monotonic ARoT persistent time.

The PP-module does not add any assumption nor OSP nor security objective of environment.

The PP-module adds a threat and corresponding security objective related to Persistent Time.

The unions of the SPD, the objectives and the security functional requirements from the base PPs and from the PP-module do not lead to a contradiction.

Furthermore, persistent time properties are independent from the functionalities defined in the ARoT Isolation PP-Module or RoT Debug PP-Module. Other PP-modules from this document can be used independently with this PP-module.

3.5.2 RoT Debug PP-Module

The RoT Debug PP-Module is intended to be used only with the base-PP of this document. It complements the RoT functionalities defined in section 2.3.1 that form the base-PP. It defines the possibility for the RoT Debug Administrator to be granted access to the Debug features, after authentication.

The PP-module does not add any assumption nor OSP nor security objective of environment.

The PP-module adds one new threat and a corresponding security objective related to access control to the debug interface. The PP-module adds six new SFRs for access control.

The unions of the SPD, the objectives and the security functional requirements from the base PPs and from the PP-module do not lead to a contradiction.

Furthermore, debug features are independent from the functionalities defined in the RoT Persistent Time PP-Module or ARoT Isolation PP-Module. Other PP-modules from this document can be used independently with this PP-module.

3.5.3 ARoT Isolation PP-Module

The ARoT Isolation PP-Module is intended to be used only with the base-PP of this document. It complements the RoT functionalities defined in section 2.3.1 that form the base-PP. It extends the isolation property defined in the base-PP to isolation between Applications Root of Trust.

The PP-module does not add any assumption nor OSP nor security objective of environment.

The PP-module adds one new threat and a corresponding security objective. The PP-module refines and replaces one SFR from the base-PP.six new SFRs for access control.

The unions of the SPD, the objectives and the security functional requirements from the base PPs and from the PP-module do not lead to a contradiction.

Furthermore, ARoT isolation properties are independent from the functionalities defined in the RoT Persistent Time PP-Module or RoT Debug PP-Module. Other PP-modules from this document can be used independently with this PP-module.

4 Security Problem Definition

This chapter introduces the security problem addressed by the RoT and its operational environment. The operational environment stands for the RoT integration and maintenance environment and the ARoT development environment. The security problem consists of the threats the RoT-enabled MCU devices may face in the field, the assumptions on its operational environment and the organizational policies that have to be implemented by the RoT or within the operational environment.

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 definitions).

4.1.1 RoT base-PP

RoT identifier

RoT identification data that is globally unique among all RoTs whatever the manufacturer, vendor or integrator. This data is typically stored in the Trusted OTP memory of the MCU or in an internal non-volatile storage dedicated to the RoT.

Properties: unique and non-modifiable.

Application Note:

The RoT identifier is intended to be public and exposed to any software running on the device, not only to Applications RoT.

ARoT code

The code of the installed Applications Root of Trust. This data is typically stored in external non-volatile memory shared with 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 RoT security services. Data and keys are owned either by the user (the owner of the RoT-enabled device) or by the ARoT service provider. This data is typically stored in external non-volatile memory shared with SPE 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 RoT reset or RoT shut-down.

Properties: monotonicity.

RoT runtime data

Runtime RoT data, including execution variables, runtime context, etc. This data is stored in volatile memory.

Properties: integrity and confidentiality, including random numbers generated by the RoT.

RoT persistent data

RoT persistent data, including 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 NSPE and potentially accessible by it.

Properties: authenticity, integrity, confidentiality and device binding.

RoT firmware

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

Properties: authenticity, integrity.

RoT initialization code and data

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

Properties: integrity.

Hardware Unique Key

The Hardware Unique Key (HUK) that is used to derive client and device specific keys, such as 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.

RoT rollback detection data

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

Properties: integrity.

4.1.2 RoT Persistent Time PP-Module

The assets of this PP-module extend the assets of the RoT base PP with a new asset "ARoT persistent time".

ARoT persistent time

Monotonic ARoT instance time persistent over RoT reset.

Properties: monotonicity.

4.1.3 RoT Debug PP-Module

The asset of this PP-module extends the assets of the RoT base PP by providing a new cryptographic key.

RoT debug authentication key

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

Properties: integrity and confidentiality

4.2 Users / Subjects

There are two kinds of users of the TOE: Applications RoT, which use the TOE services through an API, and the Non-Secure Processing Environment, which uses the TOE's services exported by the RoT and ARoT.

4.2.1 RoT base-PP

Application RoT (ARoT)

All the Applications Root of Trust running on the SPE are users of the TOE, through the 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 Applications Root of Trust, is a user of the TOE.

4.2.2 RoT Debug PP-Module

RoT Debug Administrator

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

4.3 Threats

This Protection Profile targets threats to the RoT assets that arise during the end-usage phase and can be achieved by hardware of software means. Attackers are individuals or organizations with remote or physical (local) access to the device embedding the RoT. The user of the device becomes a potential attacker when the RoT holds assets of third parties. The motivations behind the attacks may be very diverse and in general are linked to the Applications Root of Trust running on the 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 RoT or ARoT services (such as network authentication) or to threaten the reputation of the device/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 RoT-enabled device have lower impact than massive attacks that reach many devices at the same time.

This Protection Profile focuses on non-destructive software attacks that can be easily widespread, for instance through the internet, and constitute a privileged vector for getting undue access to RoT assets without damaging the device itself. 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 it, 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 analyzers, in-circuit emulators, or JTAG debuggers, which allow the attacker to operate at the package interface on the PCB. However, it is not expected that the available attack potential 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 alternatively makes a downloadable tool that is very convenient to end-users. The attacker retrieves details of the vulnerability identified in the identification phase and outputs such as attack code/executable provided by the identifier. 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

- Basic device attacker: This exploitation profile has physical access to the target device; it is the end-user or someone on his behalf. The attacker retrieves attack code/application from the identifier, guidelines written on the internet on how to perform the attack, downloads and uses tools to root or reflash the device in order to get privileged access to the SPE allowing the execution of the exploit.

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

The "threats" statement provides the general goal, the assets threatened 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 RoT base-PP

The following threats apply to any MCU RoT.

T.ABUSE_FUNCT

An attacker accesses RoT functionalities outside of their expected availability range thus violating irreversible phases of the RoT life cycle or state machine.

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

Assets threatened directly: RoT initialization code and data (integrity), RoT runtime data (confidentiality, integrity), ARoT code (authenticity, 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 authorized entities or exploiting badly implemented reset functionalities that provides undue privileges.

T.CLONE

An attacker manages to copy 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), RoT identifier (authenticity, integrity).

T.FLASH_DUMP

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

Assets threatened directly (confidentiality, authenticity, integrity): ARoT data and keys, 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 of unsoldering a flash memory and dumping its content, revealing a secret key that provides privileged access to many devices of the same model.

T.IMPERSONATION

An attacker impersonates an Application Root of Trust to gain unauthorized access to the services and data of another Application Root of Trust.

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

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

T.ROGUE_CODE_EXECUTION

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

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

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

Application Note:

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

T.PERTURBATION

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

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

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

Application Note:

Unauthorized 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 gaining system privileges) or fault injection (perturbation of the execution flow through voltage or clock glitches for instance) are examples of attack paths. The RoT can also be attacked through NSPE or ARoT "programmer errors" that exploit e.g. multi-threading or context/session management or closed sessions or by triggering system resets during execution of commands by the RoT.

T.RAM

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

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

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

Application Note:

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

T.RNG

An attacker obtains information in an unauthorized manner about random numbers generated by the RoT. This may occur for instance by a lack of entropy of the random numbers generated by the product, 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): secrets derived from random numbers.

T.SPY

An attacker discloses confidential data or keys by means of runtime attacks or unauthorized 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. shared secret key).

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

T.ROT_FIRMWARE_DOWNGRADE

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

Assets threatened directly (integrity): RoT firmware.

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

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 behavior from the storage security mechanisms. The ultimate goal of the attack is to disclose and/or modify RoT or ARoT data and/or code.

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

Application Note:

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

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, RoT persistent data, ARoT code.

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

Application Note:

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

4.3.2 RoT Persistent Time PP-Module

The following two threats apply to 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: ARoT data and keys (confidentiality, integrity).

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 RoT Debug PP-Module

T.ABUSE_DEBUG

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

Assets threatened directly: RoT initialization code and data (integrity), RoT runtime data (confidentiality, integrity), ARoT code (authenticity, integrity), 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 RoT debug mode.

4.3.4 ARoT Isolation PP-Module

T.ABUSE_AROT

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

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

4.4 Organizational Security Policies

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

4.4.1 RoT base-PP

The following policies apply to any MCU RoT.

OSP.INTEGRATION_CONFIGURATION

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

OSP.SECRETS

Generation, storage, distribution, destruction, injection of secret data in the RoT or any other operation performed outside the RoT shall enforce integrity and confidentiality of these data. This applies to secret data injected before end-usage phase (such as the HUK) or during the end-usage phase (such as cryptographic private or symmetric keys, confidential data).

OSP.UNIQUE_ROT_ID

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

4.4.2 RoT Persistent Time PP-Module

There is no additional policy in the Persistent Time PP-Module.

4.4.3 RoT Debug PP-Module

The OSP.INTEGRATION_CONFIGURATION and OSP.SECRETS organizational security policies from the base-PP are extended to also cover secure configuration of debug features and management of the secret used to authenticate the RoT Debug Administrator.

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

4.5 Assumptions

This section states the assumptions that hold on the RoT operational environment. These assumptions have to be met by the operational environment.

4.5.1 RoT base-PP

The following assumptions hold on the RoT operational environment.

A.PROTECTION_AFTER_DELIVERY

It is assumed that the TOE is protected by the environment after TOE delivery (see Section 2.5) and before entering the final usage phase (Phase 6 of the reference life-cycle). It is assumed that the persons manipulating the TOE in the operational environment apply the 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 guides, and the persons involved in delivery and protection of the product have the required skills and are aware of the security issues.

Application Note:

The certificate is 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.

The security target shall reference the applicable RoT guidelines, in particular the operational guidance that fulfills AGD_OPE.1 requirement.

A.AROT_DEVELOPMENT

ARoT developers are assumed to comply with the ARoT development guidelines set by the RoT provider. In particular, ARoT developers are assumed to consider the following principles during the development of the Applications Root of Trust:

- CA identifiers are generated and managed by the NSPE, outside the scope of the RoT. A 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 are 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.

4.5.2 RoT Persistent Time PP-Module

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

4.5.3 RoT Debug PP-Module

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

5 Security Objectives

5.1 Security Objectives for the TOE

This section states the security objectives for the RoT.

5.1.1 RoT base-PP

The following security objectives apply to any MCU RoT.

O.CA_AROT_IDENTIFICATION

The RoT 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 Applications Root of Trust.

Application Note:

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

- The Client identity of Applications Root of Trust 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 Trusted OS 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 Trusted OS
- 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.KEYS_USAGE

The RoT shall enforce on cryptographic keys the usage restrictions set by their creators.

O.ROT_ID

The RoT shall ensure that the 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:

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

O.INITIALIZATION

The RoT shall be started through a secure initialization process that ensures:

- the integrity of the RoT initialization code and data used to load the RoT firmware
- the authenticity of the RoT firmware

and that the RoT is bound to the SoC of the device. In particular, the RoT shall protect the RoT firmware against downgrade attacks.

Application Note:

The fact that the process is bound to the MCU means that the HUK cannot be modified or tampered with.

O.INSTANCE_TIME

The RoT 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.OPERATION

The RoT shall ensure the correct operation of its security functions. In particular, the RoT 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 the access to its services by the RoT and ARoTs: The RoT shall check the validity of any operation requested from either the RoT or an ARoT, at any entry point into the 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. The NSPE may be harmful but the implementation (RoT) still guarantees the stability and security of RoT. In any case, an Application Root of Trust MUST NOT be able to use a programmer error on purpose to circumvent the security boundaries enforced by an implementation
- Entry points: Software in the NSPE must not be able to call directly to RoT Functions. The NSPE software must go through protocols such that the SPM or Applications Root of Trust performs the verification of the acceptability of the operation that the NSPE software has requested.

O.RNG

The RoT 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 RoT identifier if this identifier is not generated outside the RoT.

O.RUNTIME_CONFIDENTIALITY

The RoT shall ensure that confidential RoT runtime data and ARoT data and keys are protected against unauthorized disclosure. In particular,

- The RoT shall not export any sensitive data, random numbers or secret keys to the NSPE

- The RoT shall grant access to sensitive data, random numbers or secret keys only to authorized ARoTs
- The RoT shall clean up sensitive resources as soon as it can determine that their values are no longer needed.

O.RUNTIME_INTEGRITY

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

O.AROT_AUTHENTICITY

The RoT shall verify code authenticity of Applications Root of Trust.

Application Note:

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

O.ROT_DATA_PROTECTION

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

O.ROT_ISOLATION

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

Application Note:

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

O.TRUSTED_STORAGE

The RoT shall provide Trusted Storage services for persistent ARoT general-purpose data and key material such that:

- The confidentiality of the stored data and keys is enforced
- The authenticity of the stored data and keys is enforced
- The integrity of each ARoT stored data and keys is enforced
- The atomicity of the operations that modify the storage is enforced.

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

O.PHYSICAL

The RoT must provide protection against manipulation of the TOE (including its software and TSF data and disclosure/reconstruction of user data while stored in protected memory areas and processed or against the disclosure of other critical information about the operation of the TOE.

This includes protection against

- manipulation of the hardware and any data,
- undetected manipulation of memory contents,
- physical probing on the chips surface

with a prior reverse-engineering to understand the design and its properties and functions.

The RoT must indicate or prevent its operation outside the normal operating conditions where reliability and secure operation has not been proven or tested. Examples of environmental conditions are voltage, clock frequency, temperature, or external energy fields.

O.ATTESTATION

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

O.ROLLBACK_PROTECTION

The RoT shall prevent unauthorized rollback by:

- monitoring integrity violation of RoT persistent data, RoT rollback detection data, ARoT data or keys, or ARoT code;
- react accordingly so that the 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.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.

The table below summarizes which security objectives relate to the assets stored in non-volatile and/or volatile memory.

Table 5-1: Assets storage

Asset	In non-volatile memory	In volatile memory
RoT identifier	O.ROT_ID	O.RUNTIME_INTEGRITY
ARoT code	O.AROT_AUTHENTICITY	O.RUNTIME_INTEGRITY
ARoT data and keys	O.TRUSTED_STORAGE	O.RUNTIME_INTEGRITY
RoT runtime data	N/A	O.RUNTIME_INTEGRITY
RoT persistent data	O.ROT_DATA_PROTECTION	O.ROT_DATA_PROTECTION
RoT firmware	O.INITIALIZATION	O.RUNTIME_INTEGRITY
RoT init code and data	O.INITIALIZATION	O.RUNTIME_INTEGRITY
Hardware Unique Key	O.INITIALIZATION	O.RUNTIME_INTEGRITY

RoT rollback detection data	O.ROLLBACK_PROTECTION	O.RUNTIME_INTEGRITY

5.1.2 RoT Persistent Time PP-Module

The following objective applies to RoTs implementing ARoT persistent time.

O.AROT_PERSISTENT_TIME

The RoT shall provide ARoT persistent time, which is persistent over RoT reset. The 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 RoT Debug PP-Module

O.DEBUG

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

5.1.4 ARoT Isolation PP-Module

O.AROT_ISOLATION

The RoT shall isolate the ARoT 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 the enforcement of the confidentiality and the integrity of the RoT.

5.2 Security Objectives for the Operational Environment

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

5.2.1 RoT base-PP

The following security objectives apply to any RoT operational environment that does not implement any additional security feature.

OE.INTEGRATION_CONFIGURATION

Integration and configuration of the RoT by the device manufacturer shall rely on guidelines defined by the RoT provider that state all the security requirements for the device manufacturer issued from the TOE evaluation.

OE.PROTECTION_AFTER_DELIVERY

The TOE shall be protected by the environment after TOE delivery (see Section 2.5) and before entering the final usage phase (Phase 6 of the reference life-cycle). The persons manipulating the TOE in the operational environment shall apply the RoT guidance (e.g. user and administrator guidance, installation documentation, personalization guide). The persons responsible for the application of the procedures contained in the guides, and the persons involved in delivery and protection of the product have the required skills and are aware of the security issues.

Application Note:

The certificate is valid only when the guides 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.

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 RoT shall enforce integrity and confidentiality of these data.

OE.AROT_DEVELOPMENT

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

- CA identifiers are generated and managed by the NSPE, outside the scope of the RoT; ARoTs do not assume that CA identifiers are genuine
- ARoTs do 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.UNIQUE_ROT_ID

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

5.2.2 RoT Persistent Time PP-Module

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

5.2.3 RoT Debug PP-Module

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

5.2.4 ARoT Isolation PP-Module

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

5.3 Security Objectives Rationale

5.3.1 Threats

5.3.1.1 RoT base-PP

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

- O.INITIALIZATION ensures that the RoT security functionality is correctly initialized
- O.OPERATION ensures correct operation of the security functionality and a proper management of failures
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data
- O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime
- O.AROT_AUTHENTICITY ensures that the authenticity of ARoT code is verified
- O.ROT_DATA_PROTECTION ensures that the data used by the RoT are authentic and consistent
- O.ROT_ISOLATION enforces the separation between the RoT and the outside (NSPE and ARoTs)
- O.KEYS_USAGE controls the usage of cryptographic keys
- OE.AROT_DEVELOPMENT enforces ARoT development principles, which are meant in particular to prevent disclosing information or performing modifications upon request of unauthorized entities.

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

- O.ROT_ID provides the unique RoT identifier means
- O.INITIALIZATION ensures that the RoT is bound to the SoC of the device
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data, in particular TSF data used to bind the RoT to the device
- O.RUNTIME_INTEGRITY prevents against unauthorized modification at runtime of security functionalities or data used to detect or prevent cloning
- O.ROT_DATA_PROTECTION prevents the RoT from using RoT data that is inconsistent or not authentic

- 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
- O.RNG ensures that the RoT identifier is in fact unique when generated inside the TOE
- O.ATTESTATION ensure that external entities can verify the claimed identity and security parameters of the TOE
- OE.UNIQUE_ROT_ID ensure uniqueness of the 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 Applications Root if Trust
- O.OPERATION ensures the verification of Client identities before any operation on their behalf
- O.RUNTIME_INTEGRITY prevents against unauthorized modification of security functionality at runtime.

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

- O.INITIALIZATION ensures that the RoT security functionality is correctly initialized and the integrity of RoT firmware
- O.OPERATION ensures correct operation of the security functionality
- O.RUNTIME_CONFIDENTIALITY covers runtime RoT data which might influence the behavior of the RoT
- O.AROT_AUTHENTICITY ensures that the authenticity of ARoT code is verified
- O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime
- O.ROT_DATA_PROTECTION covers persistent RoT data which might influence the behavior of the RoT
- 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-user phase
- OE.PROTECTION_AFTER_DELIVERY covers the import of foreign code in a phase other than the end-user phase.

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

- O.INITIALIZATION ensures that the RoT security functionality is correctly initialized
- O.INSTANCE_TIME ensures the reliability of instance time stamps
- O.OPERATION ensures correct operation of the security functionality and a proper management of failures
- O.RUNTIME_CONFIDENTIALITY covers runtime RoT data which might influence the behavior of the RoT
- O.AROT_AUTHENTICITY ensures that the authenticity of ARoT code is verified

- O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime
- O.AROT_ISOLATION ensures the separation of the ARoT
- O.ROT_DATA_PROTECTION covers persistent RoT data which might influence the behavior of the RoT
- O.ROT_ISOLATION enforces the separation between the RoT and the outside (NSPE and ARoTs).
- O.PHYSICAL enforces protection against physical probing and manipulation.
- Additional rationale specific to the Persistent Time PP-Module:
- O.AROT_PERSISTENT_TIME ensures the reliability of persistent time stamps

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

- O.INITIALIZATION ensures that the RoT security functionality is correctly initialized and that the initialization process is protected from the NSPE
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data at runtime
- O.RUNTIME_INTEGRITY protects against unauthorized modification of code and data at runtime
- O.AROT_ISOLATION provides a memory barrier between ARoTs
- O.ROT_ISOLATION provides a memory barrier between the RoT and the NSPE.

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

- O.INITIALIZATION ensures the correct initialization of the RoT 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 unauthorized modification, for instance to force the output of the RNG.

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

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

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

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

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

- O.OPERATION ensures the correct operation of the RoT security functionality, including storage

- O.ROT_DATA_PROTECTION ensures that stored RoT data are genuine and consistent
- O.TRUSTED_STORAGE enforces detection of corruption of the ARoT's storage
- O.AROT_AUTHENTICITY ensures that the authenticity of ARoT code is verified
- O.INITIALIZATION ensures that the firmware that is executed is the version that was intended.

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

5.3.1.2 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 RoT Debug PP-Module

T.ABUSE_DEBUG The following objective ensures protection against abuse of debug functionality:

- O.DEBUG ensure authentication of RoT Debug Administrator before granting access to RoT debug features.

5.3.1.4 ARoT Isolation PP-Module

T.ABUSE_AROT The following objective ensures protection against abuse of ARoT functionalities:

- O.AROT_ISOLATION ensure isolation between Applications Root of Trust.

5.3.2 Organizational Security Policies

5.3.2.1 RoT base-PP

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

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

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

5.3.3 Assumptions

5.3.3.1 RoT base-PP

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

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

5.3.4 SPD and Security Objectives

Table 5-2: Threats and Security Objectives - Coverage

Threats	Security Objectives	Rationale
T.ABUSE_FUNCT	O.INITIALIZATION, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.ROT_DATA_PROTECTION, O.ROT_ISOLATION, OE.AROT_DEVELOPMENT, O.KEYS_USAGE, O.AROT_AUTHENTICITY	Section 2.3.1
T.CLONE	O.ROT_ID, O.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.ROT_DATA_PROTECTION, O.TRUSTED_STORAGE, O.RNG, OE.UNIQUE_ROT_ID	Section 2.3.1
T.FLASH_DUMP	O.TRUSTED_STORAGE	Section 2.3.1
T.IMPERSONATION	O.CA_AROT_IDENTIFICATION, O.OPERATION, O.RUNTIME_INTEGRITY	Section 2.3.1
T.ROGUE_CODE_EXECUTION	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.ROT_DATA_PROTECTION, O.TRUSTED_STORAGE, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY, O.AROT_AUTHENTICITY, O.INITIALIZATION	Section 2.3.1
T.PERTURBATION	O.INITIALIZATION, O.INSTANCE_TIME, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.AROT_ISOLATION, O.AROT_PERSISTENT_TIME, O.ROT_DATA_PROTECTION, O.ROT_ISOLATION, O.AROT_AUTHENTICITY, O.PHYSICAL	Section 2.3.1
T.RAM	O.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.AROT_ISOLATION, O.ROT_ISOLATION	Section 2.3.1

Threats	Security Objectives	Rationale
T.RNG	O.INITIALIZATION, O.RNG, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY	Section 2.3.1
T.SPY	O.RUNTIME_CONFIDENTIALITY, O.AROT_ISOLATION, O.ROT_ISOLATION, O.TRUSTED_STORAGE	Section 2.3.1
T.ROT_FIRMWARE_DOWNGRADE	O.INITIALIZATION, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY	Section 2.3.1
T.STORAGE_CORRUPTION	O.OPERATION, O.ROT_DATA_PROTECTION, O.TRUSTED_STORAGE, O.AROT_AUTHENTICITY, O.INITIALIZATION	Section 2.3.1
T.ROLLBACK	O.ROLLBACK_PROTECTION	Section 2.3.1
T.AROT_PERSISTENT_TIME	O.AROT_PERSISTENT_TIME	Section 2.3.1
T.ABUSE_DEBUG	O.DEBUG	Section 2.3.1

Table 5-3: Security Objectives and Threats - Coverage

Security Objectives	Threats
O.CA_AROT_IDENTIFICATION	T.IMPERSONATION
O.KEYS_USAGE	T.ABUSE_FUNCT
O.ROT_ID	T.CLONE
O.INITIALIZATION	T.ABUSE_FUNCT, T.CLONE, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.RAM, T.RNG, T.ROT_FIRMWARE_DOWNGRADE, T.STORAGE_CORRUPTION
O.INSTANCE_TIME	T.PERTURBATION
O.OPERATION	T.ABUSE_FUNCT, T.IMPERSONATION, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.STORAGE_CORRUPTION
O.RNG	T.CLONE, T.RNG

O.RUNTIME_CONFIDENTIALITY	T.ABUSE_FUNCT, T.CLONE, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.RAM, T.RNG, T.SPY
O.RUNTIME_INTEGRITY	T.ABUSE_FUNCT, T.CLONE, T.IMPERSONATION, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.RAM, T.RNG
O.AROT_AUTHENTICITY	T.ABUSE_FUNCT, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.STORAGE_CORRUPTION
O.AROT_ISOLATION	T.PERTURBATION, T.RAM, T.SPY
O.ROT_DATA_PROTECTION	T.ABUSE_FUNCT, T.CLONE, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.STORAGE_CORRUPTION
O.ROT_ISOLATION	T.ABUSE_FUNCT, T.PERTURBATION, T.RAM, T.SPY
O.TRUSTED_STORAGE	T.CLONE, T.FLASH_DUMP, T.ROGUE_CODE_EXECUTION, T.SPY, T.STORAGE_CORRUPTION
O.PHYSICAL	T.PERTURBATION
O.ATTESTATION	T.CLONE
O.ROLLBACK_PROTECTION	T.STORAGE_CORRUPTION, T.ROLLBACK
O.AROT_PERSISTENT_TIME	T.PERTURBATION, T.AROT_PERSISTENT_TIME
O.DEBUG	T.ABUSE_DEBUG
OE.INTEGRATION_CONFIGURATION	T.ROGUE_CODE_EXECUTION, T.ROT_FIRMWARE_DOWNGRADE
OE.PROTECTION_AFTER_DELIVERY	T.ROGUE_CODE_EXECUTION, T.ROT_FIRMWARE_DOWNGRADE
OE.SECRETS	
OE.UNIQUE_ROT_ID	T.CLONE
OE.AROT_DEVELOPMENT	T.ABUSE_FUNCT

Table 5-4: OSPs and Security Objectives - Coverage

Organizational Security Policies	Security Objectives	Rationale
OSP.INTEGRATION_CONFIGURATION	OE.INTEGRATION_CONFIGURATION	Section 2.3.2
OSP.SECRETS	OE.SECRETS	Section 2.3.2
OSP.UNIQUE_ROT_ID	OE.UNIQUE_ROT_ID	Section 2.3.2

Table 5-5: Security Objectives and OSPs - Coverage

Security Objectives	Organizational Security Policies
O.CA_AROT_IDENTIFICATION	
O.KEYS_USAGE	
O.ROT_ID	
O.INITIALIZATION	
O.INSTANCE_TIME	
O.OPERATION	
O.RNG	
O.RUNTIME_CONFIDENTIALITY	
O.RUNTIME_INTEGRITY	
O.AROT_AUTHENTICITY	
O.AROT_ISOLATION	
O.ROT_DATA_PROTECTION	
O.ROT_ISOLATION	
O.TRUSTED_STORAGE	
O.PHYSICAL	
O.ATTESTATION	
O.ROLLBACK_PROTECTION	
O.AROT_PERSISTENT_TIME	
O.DEBUG	
OE.INTEGRATION_CONFIGURATION	OSP.INTEGRATION_CONFIGURATION
OE.PROTECTION_AFTER_DELIVERY	
OE.SECRETS	OSP.SECRETS
OE.UNIQUE_ROT_ID	OSP.UNIQUE_ROT_ID

Security Objectives	Organizational Security Policies
OE.AROT_DEVELOPMENT	

Table 5-6: Assumptions and Security Objectives for the Operational Environment - Coverage

Assumptions	Security Objectives for the Operational Environment	Rationale
A.PROTECTION_AFTER_DELIVERY	OE.PROTECTION_AFTER_DELIVERY	Section 2.3.3
A.AROT_DEVELOPMENT	OE.AROT_DEVELOPMENT	Section 2.3.3

Table 5-7: Security Objectives for the Operational Environment and Assumptions - Coverage

Security Objectives for the Operational Environment	Assumptions
OE.INTEGRATION_CONFIGURATION	
OE.PROTECTION_AFTER_DELIVERY	A.PROTECTION_AFTER_DELIVERY
OE.SECRETS	
OE.AROT_DEVELOPMENT	A.AROT_DEVELOPMENT

6 Extended Requirements

6.1 Extended Family FCS_RNG - Generation of random numbers

6.1.1 Objectives

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.

6.1.2 Extended Component FCS_RNG.1

Generation of random numbers requires that random numbers meet a defined quality metric.

Management: No management activities are foreseen.

Audit: 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 Extended Family FPT_INI - TSF initialization

6.2.1 Objectives

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 initialization of the TSF by a dedicated function of the TOE that ensures the initialization in a correct and secure operational state.

The family TSF Initialization (FPT_INI) is specified as follows.

6.2.2 Extended Component FPT_INI.1

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

Management: No management activities are foreseen.

Audit: No actions are defined to be auditable.

FPT_INI.1 TSF initialization

Hierarchical to: No other components.

Dependencies: No dependencies.

FPT_INI.1.1 The TOE initialization function shall verify

- the integrity of RoT initialization code and data
- the authenticity and integrity of RoT firmware
- the integrity of the HUK
- the integrity of the RoT identifier
- the version of the firmware to prevent downgrade to previous versions
- [assignment: list of implementation-dependent verifications]

prior to establishing the TSF in a secure initial state.

FPT_INI.1.2 The TOE initialization function shall detect and respond to errors and failures during initialization such that the TOE either successfully completes initialization or is halted.

FPT_INI.1.3 The TOE initialization function shall not be able to arbitrarily interact with the TSF after TOE initialization completes.

6.3 Extended family AVA_VAN_AP - Vulnerability analysis

6.3.1 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 unauthorized 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 to require parts of the implementation representation and attack potential higher than Basic.

6.3.2 Extended Component AVA_VAN_AP.3

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

AVA_VAN_AP.3 RoT 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 introduces the security problem addressed by the RoT and its operational environment. The operational environment stands for the RoT integration and maintenance environment and the ARoT development environment. The security problem consists of the threats the RoT-enabled MCU devices may face in the field, the assumptions on its operational environment and the organizational policies that have to be implemented by the RoT or within the operational environment.

7.1 Security Functional Requirements

This chapter provides the set of Security Functional Requirements (SFRs) the TOE has to enforce in order to fulfill the security objectives.

7.1.1 RoT base-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 Audit event storage
- FCS_COP.1 Cryptographic operation
- FCO_NRO.1 Selective proof of origin
- FIA_ATD.1 User attribute definition
- FIA_UID.1 Timing of identification
- FIA_UID.2 User identification before any action
- FIA_USB.1 User-subject binding
- 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
- FMT_MSA.1 Management of security attributes
- FMT_MSA.3 Static attribute initialization
- FMT_SMR.1 Security roles
- FMT_SMF.1 Management functions

- FPT_FLS.1 Failure with preservation of secure state
- FPT_ITT.1 Basic internal TSF data transfer protection
- FPT_PHP.3 Resistance to physical attacks
- 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 initialization

The statement of the security functional requirements relies on the following characterization of the 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).

Users stand for entities outside the TOE:

- Client Applications (CA), with security attribute "CA_identity" (CA identifier)
- Applications Root of Trust (ARoT), with security attribute "ARoT_identity" (ARoT identifier), "ARoT_properties".

Subjects stand for active entities inside the TOE:

- 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)
- S.API: the RoT Internal API, with security attributes "caller" (AROT identifier)
- S.RESOURCE: any software or hardware component which may be used alternatively by the 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 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 RoT and its ARoTs.

Objects stand for passive entities 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" (RoT identifier).
- OB.HUK (TSF data): the HUK, with security attribute "RoT_identity" (RoT identifier).

Cryptographic objects are a special kind of RoT objects:

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

Information stands for 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. Stands for 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 TSF persistent data (also called RoT persistent data) necessary to provide the security services. It includes all the security attributes of users, subjects, objects and information.

Cryptographic operations on user keys performed by S.API on behalf of ARoT_INSTANCE:

- OP.USE_KEY: any cryptographic operation that uses a key
- OP.EXTRACT_KEY: any operation that populates a key.

Trusted Storage operations performed by S.API on behalf of 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). It stands for object creation, object deletion, object renaming, object truncation and write to an object.

Other operations:

- Any operation executed by the RoT on behalf of an 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 ensure 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
- SFR instances: FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_ITT.1/Runtime.

ARoT Keys Access Control SFP:

- Purpose: To control the 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 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, 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, FMT_SMF.1.

Application note: The Security Target writer 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 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

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 lifespan 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, limited further by the ARoTs presence in the system

- ARoT_properties: The lifetime of this attribute is that of the availability of the ARoT to clients, limited further by the ARoTs presence in the system.

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.

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 initialization**
- **[assignment: rules for the changing of attributes].**

FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles

- **TSF**
- **ARoT_User**
- **[assignment: the authorized 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.

7.1.1.2 Confidentiality, Integrity and Isolation

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.

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 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 authorize 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 RoT subject unless one of the conditions stated in FDP_IFF.1.1/1.2/1.3/1.4 holds**.

Application Note:

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

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 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).

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: **RoT and ARoT runtime objects**.

Application Note:

This operation applies in particular upon:

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

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 RoT may reside in "physically separated parts".

7.1.1.3 Cryptography

FCS_COP.1 Cryptographic operation

FCS_COP.1.1 The TSF shall perform **[assignment: list of cryptographic operations]** 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]**.

Application Note:

The Security Target shall provide in this SFR cryptographic operations used within the TOE for:

- verifying the authenticity of RoT firmware and ARoT code
- protecting the integrity and confidentiality of Trusted Storage data. These operations are based on the HUK. The ST writer may choose to include in other iteration of FCS_COP.1 additional cryptographic operations, for instance operations provided through API to the ARoT
- performing measurements for software components and signing attestation reports.

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 RoT
- Objects: OB.AROT_KEY
- Operations: OP.USE_KEY, OP.EXTRACT_KEY.

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 authorize access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly authorize 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 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:

- This requirement states access conditions to keys through the RoT Internal API only: OP.USE_KEY and OP.EXTRACT_KEY stand for operations of the API.
- FDP_ACF.1.3/ARoT_keys: Note that ownership in the current RoT internal API specification is limited to each ARoT having access to all, and only to, its own objects.

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_KEYS.isExtractable** and **OB.AROT_KEY.owner** to the following roles:

- **change_default, query and modify OB.AROT_KEY.usage to ARoT_User role**
- **query OB.AROT_KEY.owner to the TSF role.**

FMT_MSA.3/ARoT_keys Static attribute initialization

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 authorized identified roles]** to specify alternative initial values to override the default values when an object or information is created.

7.1.1.4 Initialization, Operation and Firmware Integrity

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 RoT data: **[assignment: associated actions]**.
- detection of RoT firmware integrity violation: **[assignment: associated actions]**.
- **[assignment: list of implementation-dependent potential security violations and associated actions]**.

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 RoT runtime data, 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 RoT runtime data, 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 RoT runtime data or RoT persistent data, the TSF shall **[assignment: action that does not depend on the compromised data]**
- Upon detection of ARoT code authenticity or integrity errors, the TSF shall **abort the execution of the ARoT instance**
- Upon detection of ARoT data or ARoT keys authenticity or integrity errors, 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.

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**
- **ARoT code, ARoT data or keys authenticity or integrity failure**
- **RoT persistent data authenticity or integrity failure**
- **RoT firmware integrity failure**
- **RoT initialization failure**
- **Unexpected commands in the current RoT state**
- **[assignment: list of types of failures in the TSF].**

Application Note:

Device binding failure occurs when (part of) the stored data has not been bound by the same RoT

The ST writer shall define the characteristics of the secure state. In particular, the transition between a failure state and the secure state shall protect RoT and user data and keys confidentiality.

FPT_INI.1 TSF initialization

FPT_INI.1.1 The TOE initialization function shall verify

- the integrity of RoT initialization code and data
- the authenticity and integrity of RoT firmware
- the integrity of the HUK
- the integrity of the RoT identifier
- the version of the firmware to prevent downgrade to previous versions
- **[assignment: list of implementation-dependent verifications]**

prior to establishing the TSF in a secure initial state.

FPT_INI.1.2 The TOE initialization function shall detect and respond to errors and failures during initialization such that the TOE either successfully completes initialization or is halted.

FPT_INI.1.3 The TOE initialization function shall not be able to arbitrarily interact with the TSF after TOE initialization completes.

Application Note:

Firmware downgrade verification has to rely on data residing on the TOE, for instance on One Time Programmable (OTP) memories or EEPROM.

FPT_PHP.3 Resistance to physical attacks

FPT_PHP.3.1 The TSF shall resist **[clock frequency, voltage, temperature tampering]** to the **[microcontroller]** by responding automatically such that the SFRs are always enforced.

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 authorized users.**

FPT_TEE.1 Testing of external entities

FPT_TEE.1.1 The TSF shall run a suite of tests **prior execution and [assignment: other conditions]** to check **the fulfillment 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 RoT Identification

FAU_SAR.1 Audit review

FAU_SAR.1.1 The TSF shall provide **all users** with the capability to read **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.

FAU_STG.1 Protected audit trail storage

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

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

Application Note:

The audit record in this SFR refer to the RoT identifier. This unique identifier is stored on the TOE before TOE delivery. It can be generated on-RoT or off-RoT (the Security Target shall precise the generation method).

This identifier shall not be modified during the end-usage phase.

The Security Target shall indicate in which type of persistent memory it is stored.

7.1.1.6 Instance Time

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

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 writer shall perform the missing operation in the elements FCS_RNG.1.1 and FCS_RNG_1.2. The ST writer 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 RoT identifier if it is generated on the RoT.

7.1.1.8 Trusted Storage

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.**

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 authorize access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly authorize 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 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].**

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.

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.

FMT_MSA.3/Trusted Storage Static attribute initialization

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.

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.

7.1.1.9 Attestation

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 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 signatur**e.

7.1.2 RoT Persistent Time PP-Module

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

- FPT_STM.1 Reliable time stamps

- FMT_MTD.1 Management of TSF data
- FMT_SMF.1 Specification of Management Functions.

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 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.

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 will only affect the persistent time value of the ARoT performing the operation.

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 will only affect the persistent time value of the ARoT performing the operation.

7.1.3 RoT Debug PP-Module

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

- FIA_UID.2 User identification before any action
- FIA_UAU.2 User authentication before any action
- FIA_UAU.6 Re-authenticating
- FIA_ATD.1 User attribute definition

- FIA_USB.1 User-subject binding
- FCS_COP.1 Cryptographic operation
- FDP_ACC.1 Subset access control
- FDP_ACF.1 Security attribute based access control
- FMT_SMR.1 Security roles.

All these SFRs defined in this section only concern the Debug functionality

The additional User introduced by this PP-module is:

- 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 RoT (attribute set before TOE delivery and not modifiable afterwards) and "authenticated" (True/False) to state whether the RoT Debug Administrator has been authenticated.

This PP-module allows debug operations performed by S.DEBUG on behalf of RoT Debug Administrator:

- OP.AUTHENTICATE: activation of the debug feature by 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 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.

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.ACTIVATE, OP.DEBUG.**

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 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 authorize access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly authorize 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]**.

FCS_COP.1/Debug Cryptographic operation

FCS_COP.1.1/Debug The TSF shall perform **authentication of the 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]**.

FMT_SMR.1/Debug Security roles

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

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

Application Note:

The 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.

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.

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**.

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 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].**

FIA_UAU.2/Debug User authentication before any action

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

FIA_UAU.6/Debug Re-authenticating

FIA_UAU.6.1/Debug The TSF shall re-authenticate the user under the conditions

- after RoT power-off
- [assignment: list of conditions under which re-authentication is required].

7.1.4 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 base-PP. It provides isolation between Applications Root 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.

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 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 authorize 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 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) for the PP which consists of the EAL 2 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 will have to 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 RoT base-PP

O.CA_AROT_IDENTIFICATION The following requirements contribute to fulfill 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 the access to services and data to authorized users only
- FIA_USB.1 enforces the association of the user identity to the active entity that acts on behalf of the user and to check that this is a valid identity.

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

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

O.ROT_ID The following requirements contribute to fulfill the objective:

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

O.INITIALIZATION The following requirements contribute to fulfill the objective:

- FPT_FLS.1 states that the RoT has to reach a secure state upon initialization or device binding failure
- FCS_COP.1 states the cryptography used to verify the authenticity of RoT firmware

- FPT_INI.1 enforces the initialization of the TSF through a secure process including the verification of the authenticity and integrity of the RoT firmware.

O.INSTANCE_TIME The following requirement fulfills the objective:

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

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

- FAU_ARP.1 states the RoT responses to potential security violations
- FDP_SDI.2 enforces the monitoring of integrity and authenticity of RoT data and ARoT, and it states the behavior 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 RoT
- FPT_FLS.1 states that abnormal operations have to lead to a secure state
- 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 RoT execution spaces
- 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.
- Rationale specific to the Debug PP-Module:
  - FDP_ACC.1/Debug, FDP_ACF.1/Debug, FMT_SMR.1/Debug, FIA_UID.2/Debug, FIA_UAU.2/Debug, FIA_UAU.6/Debug, FIA_ATD.1/Debug and FIA_USB.1/Debug state the debug access policy, which grants access to the debug facilities of the RoT if this feature is not disabled

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

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

- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime ensure read access to authorized entities only
- FDP_ITT.1/Runtime and FPT_ITT.1/Runtime ensure protection against disclosure of 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 fulfill the objective:

- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state RoT and ARoT runtime data policy, which grants write access to authorized entities only
- FDP_ITT.1/Runtime and FPT_ITT.1/Runtime ensure protection against modification of 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.AROT_AUTHENTICITY The following requirements contribute to fulfill the objective:

- 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
- FCS_COP.1 states the cryptography used to verify the authenticity of ARoT code

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

- 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
- FCS_COP.1 state the cryptographic algorithms used for Trusted Storage to ensure confidentiality and authenticity of ARoT data
- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state the policy for controlling access to ARoT execution space
- FPT_FLS.1 enforces ARoT isolation by maintaining a secure state, in particular in case of panic states.

O.ROT_DATA_PROTECTION The following requirements contribute to fulfill the objective:

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

O.ROT_ISOLATION The following requirements contribute to fulfill the objective:

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

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

- FCS_COP.1 states the cryptography used to protect integrity and confidentiality of the ARoT data in external memory, if applicable
- 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 state Storage state the policy for accessing ARoT trusted storage and protecting the confidentiality of data
- FDP_SDI.2 enforces the integrity and authenticity of the trusted storage
- FPT_INI.1 enforces the integrity of RoT identifier and HUK, and it states the behavior in case of failure
- FDP_ITT.1/Trusted Storage ensure protection against disclosure of RoT and ARoT data that is transferred between resources
- FPT_FLS.1 maintains a secure state.

O.PHYSICAL The following requirements contribute to fulfill the objective:

- FPT_FLS.1 maintains a secure state
- FPT_PHP.3 ensures protection against physical attacks

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

- FCS_COP.1 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.ROLLBACK_PROTECTION The following requirements contribute to fulfill the objective:

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

7.3.1.2 RoT Persistent Time PP-Module

O.AROT_PERSISTENT_TIME The following requirements fulfill the objective:

- FPT_STM.1/Persistent Time states the persistent time reliability conditions expected from the RoT
- 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.

7.3.1.3 RoT Debug PP-Module

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

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

7.3.1.4 ARoT Isolation PP-Module

O.ABUSE_AROT The following requirements contribute to fulfill the objective:

- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime_ARoT ensure read or write access to authorized entities only
- FPT_FLS.1 maintains a secure state
- FDP_ITT.1/Trusted Storage ensure protection against disclosure of ARoT data that is transferred between resources

7.3.2 Rationale tables of Security Objectives and SFRs

Table 7-1: Security Objectives and SFRs - Coverage

Security Objectives	Security Functional Requirements	Rationale
O.CA_AROT_IDENTIFICATION	FIA_ATD.1, FIA_UID.2, FIA_USB.1	Section 4.3.1
O.KEYS_USAGE	FDP_ACC.1/ARoT_keys, FDP_ACF.1/ARoT_keys, FMT_MSA.1/ARoT_keys, FMT_MSA.3/ARoT_keys, FMT_SMF.1, FCS_COP.1, FMT_SMR.1	Section 4.3.1
O.ROT_ID	FAU_SAR.1, FCS_RNG.1, FPT_INI.1, FAU_STG.1	Section 4.3.1
O.INITIALIZATION	FPT_FLS.1, FPT_INI.1, FCS_COP.1	Section 4.3.1

Security Objectives	Security Functional Requirements	Rationale
O.INSTANCE_TIME	FPT_STM.1/Instance time	Section 4.3.1
O.OPERATION	FAU_ARP.1, FDP_SDI.2, FIA_ATD.1, FIA_UID.2, FIA_USB.1, FMT_SMR.1, FPT_FLS.1, FDP_ACC.1/Debug, FDP_ACF.1/Debug, FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FMT_SMF.1, FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FDP_ACC.1/ARoT_keys, FDP_ACF.1/ARoT_keys, FMT_MSA.1/ARoT_keys, FMT_MSA.3/ARoT_keys, FMT_SMR.1/Debug, FIA_UID.2/Debug, FIA_ATD.1/Debug, FIA_USB.1/Debug, FIA_UAU.2/Debug, FIA_UAU.6/Debug	Section 4.3.1
O.RNG	FCS_RNG.1	Section 4.3.1
O.RUNTIME_CONFIDENTIALITY	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_ITT.1/Runtime, FDP_RIP.1/Runtime, FPT_ITT.1/Runtime	Section 4.3.1
O.RUNTIME_INTEGRITY	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_ITT.1/Runtime, FPT_ITT.1/Runtime, FDP_SDI.2	Section 4.3.1
O.AROT_AUTHENTICITY	FDP_SDI.2, FCS_COP.1, FPT_TEE.1	Section 4.3.1
O.AROT_ISOLATION	FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FMT_SMF.1, FCS_COP.1, FPT_FLS.1	Section 4.3.1
O.ROT_DATA_PROTECTION	FDP_SDI.2, FCS_COP.1, FPT_ITT.1/Runtime	Section 4.3.1
O.ROT_ISOLATION	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime	Section 4.3.1
O.TRUSTED_STORAGE	FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FDP_SDI.2, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FCS_COP.1, FPT_INI.1, FMT_SMF.1, FPT_FLS.1, FDP_ITT.1/Trusted Storage	Section 4.3.1
O.PHYSICAL	FPT_FLS.1, FPT_PHP.3	Section 4.3.1
O.ATTESTATION	FIA_UID.2, FCS_COP.1, FCO_NRO.1/Attestation	Section 4.3.1
O.ROLLBACK_PROTECTION	FDP_SDI.2, FPT_FLS.1,	Section 4.3.1

Security Objectives	Security Functional Requirements	Rationale
O.AROT_PERSISTENT_TIME	FPT_STM.1/Persistent Time, FMT_MTD.1/Persistent Time, FMT_SMF.1/Persistent Time	Section 4.3.1
O.DEBUG	FDP_ACC.1/Debug, FDP_ACF.1/Debug, FCS_COP.1/Debug, FMT_SMR.1/Debug, FIA_UID.2/Debug, FIA_ATD.1/Debug, FIA_USB.1/Debug, FIA_UAU.2/Debug, FIA_UAU.6/Debug	Section 4.3.1
O.ABUSE_AROT	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime_ARoT, FDP_ITT.1/Trusted Storage, FPT_FLS.1	Section 4.3.1

Table 7-2: SFRs and Security Objectives

Security Functional Requirements	Security Objectives
FIA_ATD.1	O.CA_AROT_IDENTIFICATION, O.OPERATION
FIA_UID.2	O.CA_AROT_IDENTIFICATION, O.OPERATION
FIA_USB.1	O.CA_AROT_IDENTIFICATION, O.OPERATION
FMT_SMR.1	O.KEYS_USAGE, O.OPERATION
FDP_IFC.2/Runtime	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.AROT_ISOLATION, O.ROT_ISOLATION
FDP_IFF.1/Runtime	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.AROT_ISOLATION, O.ROT_ISOLATION
FDP_ITT.1/Runtime	O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY
FDP_RIP.1/Runtime	O.RUNTIME_CONFIDENTIALITY
FPT_ITT.1/Runtime	O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.ROT_DATA_PROTECTION
FCS_COP.1	O.KEYS_USAGE, O.INITIALIZATION, O.AROT_AUTHENTICITY, O.AROT_ISOLATION, O.ROT_DATA_PROTECTION, O.TRUSTED_STORAGE
FDP_ACC.1/ARoT_keys	O.KEYS_USAGE, O.OPERATION
FDP_ACF.1/ARoT_keys	O.KEYS_USAGE, O.OPERATION
FMT_MSA.1/ARoT_keys	O.KEYS_USAGE, O.OPERATION
FMT_MSA.3/ARoT_keys	O.KEYS_USAGE, O.OPERATION

Security Functional Requirements	Security Objectives
FAU_ARP.1	O.OPERATION
FDP_SDI.2	O.OPERATION, O.RUNTIME_INTEGRITY, O.AROT_AUTHENTICITY, O.ROT_DATA_PROTECTION, O.TRUSTED_STORAGE, O.ROLLBACK_PROTECTION
FPT_FLS.1	O.INITIALIZATION, O.OPERATION, O.AROT_ISOLATION, O.TRUSTED_STORAGE, O.ROLLBACK_PROTECTION, O.PHYSICAL, O.ABUSE_AROT
FPT_INI.1	O.ROT_ID, O.INITIALIZATION, O.TRUSTED_STORAGE
FPT_PHP.3	O.PHYSICAL
FMT_SMF.1	O.KEYS_USAGE, O.OPERATION, O.AROT_ISOLATION, O.TRUSTED_STORAGE
FPT_TEE.1	O.AROT_AUTHENTICITY
FAU_SAR.1	O.ROT_ID
FAU_STG.1	O.ROT_ID
FPT_STM.1/Instance time	O.INSTANCE_TIME
FCS_RNG.1	O.ROT_ID, O.RNG
FDP_ACC.1/Trusted Storage	O.OPERATION, O.AROT_ISOLATION, O.TRUSTED_STORAGE
FDP_ACF.1/Trusted Storage	O.OPERATION, O.AROT_ISOLATION, O.TRUSTED_STORAGE
FDP_ROL.1/Trusted Storage	O.TRUSTED_STORAGE
FMT_MSA.1/Trusted Storage	O.OPERATION, O.AROT_ISOLATION, O.TRUSTED_STORAGE
FMT_MSA.3/Trusted Storage	O.OPERATION, O.AROT_ISOLATION, O.TRUSTED_STORAGE
FDP_ITT.1/Trusted Storage	O.TRUSTED_STORAGE, O.ABUSE_AROT
FPT_STM.1/Persistent Time	O.AROT_PERSISTENT_TIME
FMT_MTD.1/Persistent Time	O.AROT_PERSISTENT_TIME
FMT_SMF.1/Persistent Time	O.AROT_PERSISTENT_TIME
FDP_ACC.1/Debug	O.OPERATION, O.DEBUG
FDP_ACF.1/Debug	O.OPERATION, O.DEBUG
FCS_COP.1/Debug	O.DEBUG
FMT_SMR.1/Debug	O.OPERATION, O.DEBUG

Security Functional Requirements	Security Objectives
FIA_UID.2/Debug	O.OPERATION, O.DEBUG
FIA_ATD.1/Debug	O.OPERATION, O.DEBUG
FIA_USB.1/Debug	O.OPERATION, O.DEBUG
FIA_UAU.2/Debug	O.OPERATION, O.DEBUG
FIA_UAU.6/Debug	O.OPERATION, O.DEBUG
FDP_IFF.1/Runtime_ARoT	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.AROT_ISOLATION, O.ROT_ISOLATION

7.3.3 Dependencies

7.3.3.1 SFRs Dependencies

Table 7-3: SFRs Dependencies

Requirements	CC Dependencies	Satisfied Dependencies
FDP_ACC.1/Debug	(FDP_ACF.1)	FDP_ACF.1/Debug
FDP_ACF.1/Debug	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/Debug
FCS_COP.1/Debug	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	
FMT_SMR.1/Debug	(FIA_UID.1)	FIA_UID.2/Debug
FIA_UID.2/Debug	No Dependencies	
FIA_ATD.1/Debug	No Dependencies	
FIA_USB.1/Debug	(FIA_ATD.1)	FIA_ATD.1/Debug
FIA_UAU.2/Debug	(FIA_UID.1)	FIA_UID.2/Debug
FIA_UAU.6/Debug	No Dependencies	
FIA_ATD.1	No Dependencies	
FIA_UID.2	No Dependencies	
FIA_USB.1	(FIA_ATD.1)	FIA_ATD.1
FMT_SMR.1	(FIA_UID.1)	FIA_UID.2
FDP_IFC.2/Runtime	(FDP_IFF.1)	FDP_IFF.1/Runtime
FDP_IFF.1/Runtime	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/Runtime
FDP_ITT.1/Runtime	(FDP_ACC.1 or FDP_IFC.1)	FDP_IFC.2/Runtime
FDP_RIP.1/Runtime	No Dependencies	
FPT_ITT.1/Runtime	No Dependencies	

Requirements	CC Dependencies	Satisfied Dependencies
FCS_COP.1	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	
FDP_ACC.1/ARoT_keys	(FDP_ACF.1)	FDP_ACF.1/ARoT_keys
FDP_ACF.1/ARoT_keys	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/ARoT_keys, FMT_MSA.3/ARoT_keys
FMT_MSA.1/ARoT_keys	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FDP_ACC.1/ARoT_keys, FMT_SMF.1
FMT_MSA.3/ARoT_keys	(FMT_MSA.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_MSA.1/ARoT_keys
FAU_ARP.1	(FAU_SAA.1)	
FDP_SDI.2	No Dependencies	
FPT_FLS.1	No Dependencies	
FPT_PHP.3	No Dependencies	
FPT_INI.1	No Dependencies	
FMT_SMF.1	No Dependencies	
FPT_TEE.1	No Dependencies	
FAU_SAR.1	(FAU_GEN.1)	
FAU_STG.1	(FAU_GEN.1)	
FPT_STM.1/Instance time	No Dependencies	
FCS_RNG.1	No Dependencies	
FDP_ACC.1/Trusted Storage	(FDP_ACF.1)	FDP_ACF.1/Trusted Storage
FDP_ACF.1/Trusted Storage	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/Trusted Storage, FMT_MSA.3/Trusted Storage
FDP_ROL.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted Storage
FMT_MSA.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_SMF.1, FDP_ACC.1/Trusted Storage
FMT_MSA.3/Trusted Storage	(FMT_MSA.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_MSA.1/Trusted Storage
FDP_ITT.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted Storage
FCO_NRO.1/Attestation	(FIA_UID.1)	FIA_UID.2
FPT_STM.1/Persistent Time	No Dependencies	

Requirements	CC Dependencies	Satisfied Dependencies
FMT_MTD.1/Persistent Time	(FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_SMF.1/Persistent Time
FMT_SMF.1/Persistent Time	No Dependencies	
FDP_IFF.1/Runtime_ARoT	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/Runtime

7.3.3.2 Rationale for the exclusion of Dependencies

The dependency FMT_MSA.3 of FDP_ACF.1/Debug is discarded. There is no management of security attributes by authorized 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.

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

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

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 authorized 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.

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

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

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.

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

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

7.3.3.3 SARs Dependencies

Table 7-4: SARs Dependencies

Requirements	CC Dependencies	Satisfied Dependencies
ADV_ARC.1	(ADV_FSP.1) and (ADV_TDS.1)	ADV_FSP.2, ADV_TDS.1

Requirements	CC Dependencies	Satisfied Dependencies
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
ALC_CMS.2	No Dependencies	
ALC_DEL.1	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 in order 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 ROT commercial development practices that are compatible with industry constraints, in particular the life cycle of ROT and ROT-enabled MCU devices. The developer has to 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 ROT and the interest that ROT-enabled MCU devices 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.