SAMSUNG

Samsung TEEgris Security Target on Exynos 2200

Author	m.sec.teesdk@samsung.com	Date/Version	2023-08-09 / doc v2.10 TEEgris v5.0.0
Organization	Security Engineering Group, Samsung Electronics		

1 Introduction

This document defines the Security Target for Samsung TEEgris (i.e., TEEgris) integrated in Samsung LSI ARMv8 and ARMv9 based Mobile SoC on which Android T runs as the rich OS.

This Security Target is conformant to the TEE Protection Profile v1.3 and is intended to apply for GlobalPlatform TEE security certification (cf. [TEE CP], [TEE EM]).

1.1 Objective

This document provides the security identification and functional requirement of TEEgris which intended to elaborate security aspects of the operating system itself and trusted application development.

1.2 References

Table 1: Normative Reference

Standard / Specification	Description	Ref
CC Part 1	Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1, revision 5, April 2017.	[CC1]
CC Part 2	Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Requirements. Version 3.1, revision 5, April 2017.	[CC2]
CC Part 3	Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements. Version 3.1, revision 5, April 2017.	[CC3]
CC Part 1	Common Methodology for Information Technology Security Evaluation, Evaluation methodology. Version 3.1, revision 5, April 2017.	[CEM]
GP_PRO_023	GlobalPlatform Device Technology TEE Certification Process (last applicable version)	[TEE CP]
GPD_GUI_044	GlobalPlatform Device Technology TEE Evaluation Methodology (last applicable version)	[TEE EM]
GPD_SPE_007	GlobalPlatform Device Technology TEE Client API Specification v1.0	[TEE CLIENT 1.0]
GPD_EPR_028	GlobalPlatform Device Technology TEE Client API Specification v1.0 Errata and Precisions v2.0	[TEE CLIENT E 2.0]
GPD_SPE_010	GlobalPlatform Device Technology TEE Internal Core API Specification v1.0	[TEE CORE 1.0]
	GlobalPlatform Device Technology TEE Internal Core API Specification v1.2.1	[TEE CORE 1.2]
GPD_EPR_017	GlobalPlatform Device Technology TEE Internal Core API Specification v1.0 Errata and Precisions v1.0	[TEE CORE E 1.0]
	GlobalPlatform Device Technology TEE Internal Core API Specification v1.0 Errata and Precisions v3.0	[TEE CORE E 3.0]
GPD_SPE_024	GlobalPlatform Device Technology TEE Secure Element API Specification v1.0	[TEE SE 1.0]
GPD_EPR_030	GlobalPlatform Device Technology TEE Secure Element API Specification v1.0 Errata and Precisions v1.0	[TEE SE E 1.0]
GPD_SPE_020	GlobalPlatform Device Technology Trusted User Interface API Specification v1.0	[TEE TUI 1.0]

Standard / Specification	Description	Ref
GPD_SPE_055	GlobalPlatform Device Technology Trusted User Interface Low-level API specification v1.0.1	[TEE TUILL 1.0]
GPD_SPE_025	GlobalPlatform Device Technology TEE TA Debug Specification v1.0	[TEE TA DEBUG 1.0]
GP_CAT	TEE Initial Configuration Test Suite v1.1.0.1	[TEE ICTS 1.1.0.1]
GPD_SPE_021	TEE Protection Profile PP-configuration composed of the base Protection Profile only v1.3	[TEE PP 1.3]
GPD_SPE_021+Time	TEE PP-configuration composed of the base Protection Profile and the TEE Time and Rollback PP-module v1.3	[TEE PP T 1.3]
GPD_SPE_021+Debug	TEE PP-configuration composed of the base Protection Profile and the TEE Debug PP-module v1.3	[TEE PP D 1.3]
GPD_SPE_021+Time&Debug	TEE PP-configuration composed of the base Protection Profile and the TEE Time and Rollback and TEE Debug PP-modules v1.3	[TEE PP TD 1.3]
GPD_SPE_142	Trusted User Interface PP-Module v1.0	[TEE PP TUI 1.0]
GPD_NOT_051	Application of Attack Potential to Trusted Execution Environment – Confidential version (Attack Catalog)	[TEE AP]
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]
IETF RFC 2119	Key words for use in RFCs to Indicate Requirement Levels	[RFC 2119]
NIST Special Publication	Recommendation for Random Number Generation Using Deterministic Random Bit Generators. NIST Special Publication 800-90A Revision 1. June 2015	[NIST 800-90A]
NIST Technical Publication	Recommendation for Key Derivation Using Pseudorandom Functions	[NIST 800-108]
ANSSI guide	Guide Des Mécanismes Cryptographiques	[ANSSI-PG-083]
OMTP ATE TR1	Open Mobile Terminal Platform Advanced Trusted Environment OMTP TR1 v1.1	[OMTP-TR1]
FIPS Publication	FIPS 180-4 – Secure Hash Signature Standard (SHS), March 2012	[Hash]
FIPS Publication	FIPS 197 – Advanced Encryption Standard, November 2001	[AES]
IEEE Standard	IEEE Std 1619-2007 – IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices, April 2008	
NIST Special Publication	NIST SP800-38A – Recommendation for Block Cipher Modes of Operation, October 2010	
RFC	RFC 1423 – Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifier, February 1993s	
FIPS Publication	FIPS 46-3 – Data Encryption Standard (DES), October 1999	[3DES]
FIPS Publication	FIPS 81 – DES Mode of Operations	
RSA Laboratories Publication	PKCS#1 – RSA Cryptographic Standard. PCKS#1 v2.2. October 2012	[RSA]
FIPS Publication	FIPS 186-2 - Digital Signature Standard (DSS), January 2000	[DSA]
FIPS Publication	FIPS 186-4 - Digital Signature Standard (DSS), July 2013	[ECDSA]
ANSI	ANSI X9.62 - Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECSDA),	

Standard / Specification	Description	Ref
NIST Special Publication	NIST SP800-56A - Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, March 2007	[ECDH]
FIPS Publication	FIPS 186-4 - Digital Signature Standard (DSS), July 2013	
RSA Laboratories Publication	PKCS#3- Diffie-Hellman Key Agreement Standard	[DH]
RFC	RFC 4231 Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, December 2005	[HMAC]
RFC	RFC 2202 - Test cases for HMAC-MD5 and HMAC-SHA-1, September 1997	
NIST Special Publication	NIST SP800-38B - Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication, May 2005	[CMAC]
RFC	RFC 7748 Elliptic Curves for Security	[X25519]
RFC	RFC 8032 Edwards-Curve Digital Signature Algorithm (EdDSA)	[ED25519]

Table 1-2. Informative References

Standard / Specification	Description	Ref
Samsung_Proprietary_API_Coverage – 2022-09-22	Samsung Internal API reference	[SS REF]
Samsung_TEEgris_GP_Internal_Core_API Coverage – 2023-03-09	Samsung TEEgris GP Internal Core API Coverage	[SS COV]
Samsung_TEEgris_Overview and Guidance – v1.5, 2023-08-09	Samsung TEEgris Overview and Guidance	[SS OGD]
Samsung TEEgris Design – v1.3, 2023-08-09	Samsung TEEgris Design	[SS TDS]
Samsung_TEEgris_Security_Architecture – v1.1, 2023-03-24	Samsung TEEgris Security Architecture	[SS ARC]
Samsung Cryptographic Specification – v1.3, 2023-08-09	Samsung TEEgris cryptographic specification	[SS CRY]

1.3 Terminology and Definitions

1.3.1 Key Words

The key words “MUST”, “MUST NOT”, “SHALL”, “SHALL NOT”, “REQUIRED”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document indicate normative statements and are to be interpreted as described in [RFC 2119].

1.3.2 Other Terminology

Selected terms used in this document are included in the above table. Additional terms are defined in the references.

Table 1-3. Terminology and Definitions

Term	Definition
Application Programming Interface (API)	A set of rules that software programs can follow to communicate with each other.

Term	Definition
Client Application (CA)	An application running outside of the Trusted Execution Environment (TEE) making use of the TEE Client API that accesses facilities provided by the Trusted Applications inside the TEE. Contrast Trusted Application.
Consistency	A property of the TEE persistent storage that stands at the same time for runtime and startup consistency. Runtime consistency stands for the guarantee that the following clauses hold: • Read/Read: Two successful readings from the same storage location give the same value if the TEE did not write to this location and the TEE was not reset in between • Write/Read: A successful reading from a given storage location gives the value that the TEE last wrote to this location if the TEE was not reset in between. Startup consistency stands for the guarantee that the following clause holds: • During a given power cycle, the stored data used at startup is the data for which runtime consistency was enforced on the same TEE on a previous power cycle. Consistency implies runtime integrity of what is successfully written and read back – values or code. However, the stored data used at startup may be restored from an old power cycle, not the latest one. It is still consistent at start-up because it corresponds to a memory snapshot at a given time, but it represents an integrity loss compared with the latest power cycle. This notion is weaker than integrity that must be preserved between power cycles.
Device binding	Device binding is the property of data being only usable on a unique given system instance, here a TEE.
Execution Environment (EE)	A set of hardware and software components that provide facilities (computing, memory management, input/out, etc.) necessary to support applications.
Monotonicity	Monotonicity is the property of a variable whose value is either always increasing or always decreasing over time.
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 TEE	A TEE residing in a device that is in the end user phase of its life cycle.
REE Communication Agent	An REE Rich OS driver that enables communication between the REE and the TEE. Contrast TEE Communication Agent.
Rich Execution Environment (REE)	An environment that is provided and governed by a Rich OS, potentially in conjunction with other supporting operating systems and hypervisors; it is outside of the TEE. This environment and applications running on it are considered un-trusted. Contrast Trusted Execution Environment.

Term	Definition
Rich OS	Typically, an OS providing a much wider variety of features than that of the OS running inside the TEE. 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 Rich OS it will run in an execution environment that may be larger than the TEE hardware (often called an REE – Rich Execution Environment) with much lower physical security boundaries. From the TEE viewpoint, everything in the REE has to be considered un-trusted, though from the Rich OS point of view there may be internal trust structures. Contrast Trusted 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.
Secure OS	Similar Trusted OS
Standard OS	Similar Rich OS
System-on-Chip (SoC)	An electronic system all of whose components are included in a single integrated circuit.
TA instance time / TA persistent time	Time value available to a Trusted Application through the TEE Internal 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 TA instance, and the returned value is called “TA instance time”. • Persistent time depends only on the TA but not on a particular instance, it must be monotonic even across power cycles. Its monotonicity across power cycles is related to the Time and Rollback optional PP-module.
TEE Client API	The software interface used by clients running in the REE to communicate with the TEE and with the Trusted Applications executed by the TEE.
TEE Communication Agent	A TEE Trusted OS driver that enables communication between REE and TEE. Contrast REE Communication Agent.
TEE Internal API	The software interface exposing TEE functionality to Trusted Applications.
TEE Service Library	A software library that includes all security related drivers.
Trusted Application (TA)	An application running inside the Trusted Execution Environment that exports security related functionality to Client Applications outside of the TEE. Contrast Client Application.
Trusted Execution Environment (TEE)	An execution environment that runs alongside but isolated from an REE. A TEE has security capabilities and meets certain security-related requirements: It protects TEE 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 TEE, and the level of security achieved varies accordingly. For more information, see OMTP ATE TR1 [OMTP-TR1]. Contrast Rich Execution Environment.

Term	Definition
Trusted OS	The operating system running in the TEE. It has been designed primarily to enable the TEE using security-based design techniques. It provides the GlobalPlatform TEE Internal API to Trusted Applications and a proprietary method to enable the GlobalPlatform TEE Client API software interface from other EE. Similar Secure OS. Contrast Rich OS.
Trusted Storage	In GlobalPlatform TEE documents, trusted storage indicates storage that is protected to at least the robustness level defined for OMTP Secure Storage (in section 5 of [OMTP-TR1]). It is protected either by the hardware of the TEE, or cryptographically by keys held in the TEE. If keys are used they are at least of the strength used to instantiate the TEE. A GlobalPlatform TEE Trusted Storage is not considered hardware tamper resistant to the levels achieved by Secure Elements.

1.4 Abbreviation and Notations

Table 1-4. Abbreviation and Notations

Abbreviation / Notation	Meaning
AES	Advanced Encryption Standard (defined in [AES])
ATF	ARM Trusted Firmware
APC	Access Permission Control
ARM	Advanced RISC (Reduced Instruction Set Computer) Machine
API	Application Programming Interface
BROM	Boot ROM
CA	Client Application
DDR RAM	Double Data Rate RAM
DES	Data Encryption Standard (defined in [DES])
DRAM	Dynamic RAM
DRM	Digital Rights Management
EE	Execution Environment
EMMC	Embedded Multi-Media Card
EPBL	Exynos Primary Bootloader
I2C	Intelligent Interface Controller
ID	IDentifier
ISP	Image Signal Processor
FM	Frequency Modulation
GPIO	General Purpose Input/Output
GPS	Global positioning System
GPU	General Processor Unit
HD	High-Definition
HDMI	High-Definition Multimedia Interface
IPsec	Internet Protocol security
JTAG	Joint Test Action Group (defined in [JTAG])
MAC	Message Authentication Code

Abbreviation / Notation	Meaning
N/A	Not Applicable
NFC	Near Field Communication
NVM	Non-Volatile Memory
OAEP	Optimal Asymmetric Encryption Padding
OMTP	Open Mobile Terminal Platform
OS	Operating System
OTP	One Time Programmable memory
PKCS	Public-Key Cryptography Standard
PP	Protection Profile (defined in [CC1])
RAM	Random Access Memory
REE	Rich Execution Environment
RFC	Request For Comments; may denote a memorandum published by the IETF
RNG	Random Number Generator
ROM	Read Only Memory
RSA	Rivest / Shamir / Adleman asymmetric algorithm (defined in [RSA])
RPMB	Reply Protected Memory Block
SD/MMC	Secure Digital/Multi-Media Card
SFP	Security Function Policy (defined in [CC1])
SFR	Security Functional Requirement (defined in [CC1])
SHA	Secure Hash Algorithm (defined in [SHA])
SIM	Subscriber Identity Module
SMC	Secure Monitor Call
SoC	System-on-Chip
SPI	Serial Peripheral Interface
SRAM	Static RAM
SSL	Secure Sockets Layer
ST	Security Target (defined in [CC1])
TA	Trusted Application
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TOE	Target of Evaluation (defined in [CC1])
TRNG	True RNG
TSF	TOE Security Functionality (defined in [CC1])
UART	Universal Asynchronous Receiver-Transmitter
UFS	Universal Flash Storage
USB	Universal Serial Bus
VPN	Virtual Private Network
WDT	Watch Dog Timer

Abbreviation / Notation	Meaning
WIFI	Wireless Fidelity

1.5 Revision History

Table 1-5. Revision History

Date	Version	Description
December 13, 2019	0.1	Initial draft version of the document.
February 03, 2020	0.2	Functional description added, few typo fixed
March 19, 2020	1.0	Version update for GP submission
May 15, 2020	1.0	Submission to GP
July 28, 2020	1.1	Revised according to GP Secretariat review
August 26, 2020	1.2	Revised according to second round of review
August 08, 2022	1.3	Initial draft version of the document, as 5.0 preparation Reflected IOTR feedback_20220801
September 20, 2022	2.0	Revised for conformance with PP TEE v1.3 and to include Trusted User Interface PP-module
October 4, 2022	2.1	Updated with Samsung answers For Thales and Samsung review
October 27, 2022	2.2	Updated following Thales and Samsung comments
January 5, 2023	2.3	Updated according to Crypto specification Updated TEE identification after evaluation
February 14, 2023	2.4	Updated according to Thales comments after review
February 21, 2023	2.5	Updated according to Thales comments after review
March 10, 2023	2.6	Updated according to Thales comments after review
March 24, 2023	2.7	Updated guide for integrator and jtag according to Thales comments after review
May 31, 2023	2.8	Updated identification and typo in referencing SFR according to Thales comments after review
July 20, 2023	2.9	Updated references
August 9, 2023	2.10	Updated references

2 Identification

This section presents how TEEgris provides the principle of its asset protection.

2.1 Security Target Identification

The following table provides Security Target identification information.

Security Target Identification	
Document title	Samsung TEEgris Security Target on Exynos 2200
Document reference	Samsung_TEEgris_Security_Target_v2.10
Document version	2.10
Document status	Initial submission/Confidential
Document author	Samsung TEEgris

2.2 TOE Identification

2.2.1 TOE Type

The following table provides TOE characterization information.

TOE characterization	
TOE Type	TEE on Final Device
Multiple TOEs	No

2.2.2 TOE References

The following table provides TOE identification information.

TOE Identification	
SoC reference	s5e9925
Device reference	Samsung Galaxy S22, S22+, S22 Ultra
Model reference	SM-S901B, SM-S906B, SM-S908B
DRAM reference	Samsung LPDDR5 8G/12G 3200MHz
Commercial name(s)	Samsung Secure OS (TEEgris) version 5.0.0.0
Main developer	Samsung Electronics
ROM/Boot code reference	Zagreb-SP1A-1727 sha256: 0cb293569ae1a33477f27b0acd58bd455c4d88bee53af3584facb2bc6e2f74c5
TEE binary reference	Samsung Secure OS Release Version 5.0.0.0 sha256: df6e524d7accacbc722776660ffd698b023a6c55ba8cbe3b0e5cacb5b1d789db
ATF binary reference	Zagreb-SP1A-2124V1-2125R1 sha256: 0f78b256b193620bca0f861b550371c14ac5e3b7c40d049fe6ff63a27d3d0512
TEE binary developer	Samsung Electronics Co., Ltd.

Guidance documents below are part of TOE.

The following table provides the identification information of main guidance document for TEE integrators:

Guidance for TEE integrators	
Document title	Samsung TEEgris Overview and Guidance
Document date	9 August, 2023
Document status	Confidential
Document author	Mobile Security TEEgris SDK - m.sec.teesdk@samsung.com
Comments	None

The following table provides the identification information of main guidance document for Trusted Applications developers:

Guidance for TA/CA developers	
Document title	Samsung TEEgris Overview and Guidance
Document date	9 August, 2023
Document status	Confidential
Document author	Mobile Security TEEgris SDK - m.sec.teesdk@samsung.com
Comments	None

The following table provides the identification information of main guidance document for TEE final users:

Guidance for TEE final users	
Document title	Samsung TEEgris Overview and Guidance
Document date	9 August, 2023
Document status	Confidential
Document author	Mobile Security TEEgris SDK - m.sec.teesdk@samsung.com
Comments	None

2.2.3 Non-TOE Component Identification

The following table lists the non-TOE components which are required for the operation of the TEE or have some relationship with the TOE:

Name	Reference	Main developer
Android	T	Google
TZ Driver (tzdev)	(part of Linux kernel)	Samsung
TZ Socket Driver (tziwsock)	(part of Linux kernel)	Samsung

The following table lists the identification information of the Trusted Applications pre-loaded in the TOE. All TA has guidance reference to [SS ARC].

Pre-loaded TA identification developed by Samsung		
TA identifier / Commercial name(s)	**Privileges**	**TA binary reference**
Keymaster	N/A	00000000-0000-0000-0000-4b45594d5354 00000000-0000-0000-0000-534258505859
Gatekeeper	Driver	00000000-0000-0000-0000-474154454b45
Key management	N/A	00000000-0000-0000-0000-000000534b4d 00000000-0000-0000-0000-505256544545 00000000-0000-0000-0000-0000534b504d
TIMA (attestation, keystore, integrity)	N/A	00000000-0000-0000-0000-00000000dead 00000000-0000-0000-0000-00535453540c 00000000-0000-0000-0000-0053545354ab
TIMA (management)	Driver	00000000-0000-0000-0000-00535453540b
Biometric (face, fingerprint)	Driver	00000000-0000-0000-0000-42494f535542 00000000-0000-0000-0000-46494e474502 00000000-0000-0000-0000-46494e474552 00000000-0000-0000-0000-5345435f4652 00000000-0000-0000-0000-465044726976
DRM (widevine, hdcp)	Driver	00000000-0000-0000-0000-00575644524d 00000000-0000-0000-0000-000048444350
Trusted user interface	Driver	00000000-0000-0000-0000-000000010081 00000000-0000-0000-0000-000000020081
Authentication	Driver	00000000-0000-0000-0000-0050524f4341 00000000-0000-0000-0000-000046495645
Key/Data management	N/A	00000000-0000-0000-0000-564c544b5052 00000000-0000-0000-0000-564c544b4456 00000000-0000-0000-0000-53626f786476
Samsung Payment	N/A	00000000-0000-0000-0000-00504159544D 00000000-0000-0000-0000-504159415554 00000000-0000-0000-0000-564953415059 00000000-0000-0000-0000-4D4153545059 00000000-0000-0000-0000-414D45585059

The above list contains representative TAs but not limited to the exhaustive list (Except for driver).

3 TOE Description

3.1 Expected Usage

The TEE enables the use of mobile devices for a wide range of services that require security protection, for instance:

- Corporate services: enterprise devices that enable push e-mail access and office applications give employees a flexibility that requires a secure and fast link to their workplace applications through Virtual Private Networking (VPN), secure storage of their data, and remote management of the device by the IT department.
- Content management: today's devices offer HD video playback and streaming, mobile TV broadcast reception, and console-quality 3-D games. This functionality often requires content protection, through Digital Rights Management (DRM) or Conditional Access.
- Personal data protection: devices store increasing amounts of personal information (such as contacts, messages, photos and video clips) and even sensitive data (credentials, passwords, health data, etc.). Secure storage means are required to prevent exposure of this information in the event of loss, theft, or any other adverse event, such as a malware.
- Connectivity protection: networking through multiple technologies—such as 3G, 4G or Wi-Fi/WiMAX, as well as personal communication means, such as Bluetooth® and Near Field Communication (NFC) — enables the use of mobile devices for peer-to-peer communication and for accessing the Internet. Such access, including web services or remote storage relying on cloud computing, typically uses SSL/TLS or IPsec internet secure protocols. Often the handling of the key material or the client end of the session needs to be secured.
- Mobile financial services: some types of financial services tend to be targeted at smart phones, such as mobile banking, mobile money transfer, mobile authentication (e.g. use with One-Time Password – OTP technology), mobile proximity payments, etc. These services require secure user authentication and secure transaction, which can be performed by the device potentially in cooperation with a Secure Element.
- (Biometric) Authentication services: Such services require a robust root-of-trust, isolation from other execution environments, and the controlled use of the user interfaces, such as biometric sensors and displays.

3.2 Overview

The TEE implementation relies on a SoC that is embedded in several final devices. In addition to SoC, hardware components should be considered to be TEE integration into final device.

3.2.1 Hardware Architecture

The TEE is composed of the following hardware components:

- Hardware processing unit (with Security Extension and 8 cores)
  - Montblanc (Matterhorn-ELP) single-core, Matterhorn (Cortex-A710) triple-core, and Klein (Cortex-A510) quad-core processor
- Physical volatile memory: **Secure DRAM** (138MB external DRAM reserved for Secure OS) and **Secure Internal SRAM**;
- Physical non-volatile memory: Secure BROM, Efuse (Secure OTP);
- Memory Protection Unit (MPU) as hardware solution for definition and isolation of memory areas;
- Secure peripherals:
  - Accessible from Secure World only: Secure JTAG, PRNG/TRNG, Secure Timer, Secure Watchdog, CryptoEngine (AES, SHA256, and RSA);

- Shared with Normal World (the hardware instance can be switched between controls under Secure and Normal Worlds depending on the current session): **SD/MMC**, **eMMC or UFS**, **USB20**, **UARTs**, **I2Cs**, **Timers**, **SPI**, **KEYPAD**, **GPIO**, **Watchdog**, **GPU**, **Video Encode**, **Video Decode**, **ISP**, and **Display Controller**.

- Connections between the processing unit(s) and the hardware resources: AXI-based Bus.

Components in **bold character** are the external hardware interfaces from which the TEE functionalities and the assets can be accessed from the TEE environment.

Some of the components can be accessed either by the Normal World or by the Secure World ("dynamically secured") at a given instance, but not simultaneously, based on a status set to "secure" or "non-secure". The TEE is in charge of securely handling the switch of this status. The TEE also ensures the access control to the hardware components depending on the associated access rights and the current status.

3.2.2 Software Architecture

From a software point of view, the TEE is embedded in the device and runs alongside a standard OS, or Rich Execution Environment (REE).

The TEE software architecture identifies two distinct classes of components:

- The Trusted Applications that run on the TEE and use the TEE Internal API;
- The Trusted OS component whose role is
  - to provide communication facilities with the REE software and the system level functionality required by the Trusted Applications, accessible from the TEE Internal API. An implementation can be either built-in or driver-alike, although TA can access the functionality transparently.
  - to operate basic operating system, supporting memory, resource, peripheral, data management.

The REE software components interacting with the TEE are of two types:

- The Client Applications which make use of the TEE Client API to access the secure services offered by TAs running on the TEE;
- The Rich OS, which provides the TEE Client API and sends requests to the TEE.

The TEE software architecture and the interactions with the software environment are illustrated on the figure below:

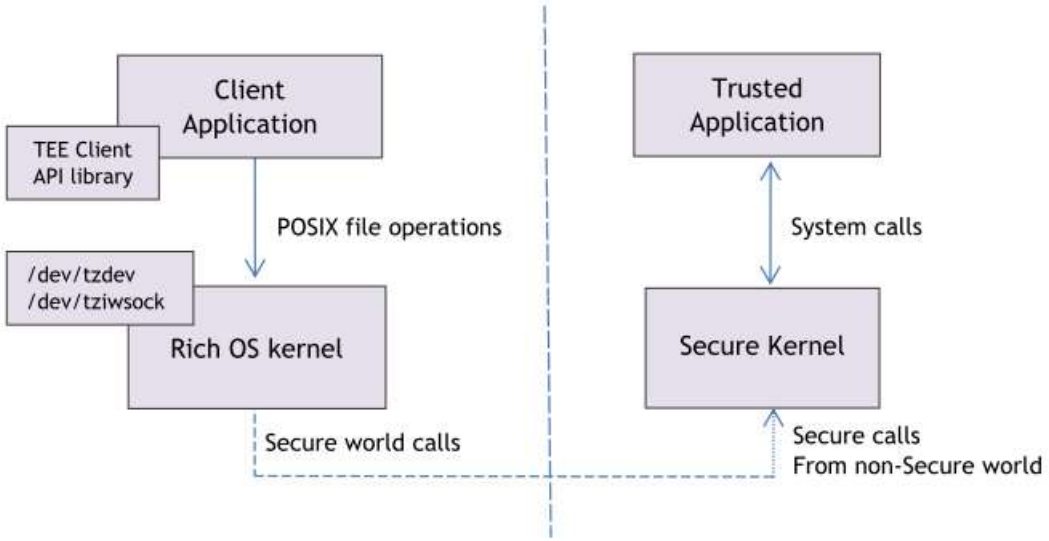

Figure 3-1 TEE overall software architecture

The TEE software external interfaces comprise:

- APIs provided to the Trusted Applications: they include the GlobalPlatform and the proprietary APIs;
- Shared memory between TEE and REE;
- TEE Communication Agent protocol to interact with the REE.

GlobalPlatform APIs:

The list of GlobalPlatform APIs implemented by the TEE is provided in documents [SS COV].

Proprietary APIs:

The TEE implements the following proprietary APIs defined in [SS REF]:

- **Loadable driver API**: provides a set of functions to interact with device drivers from user space components.
- **SPI API**: provides a set of functions to manipulate device connected with SPI bus.
- **I2C API**: provides a set of functions to manipulate device connected with I2C bus.
- **Trusted User Interface API**: provides a set of functions to manipulate low-lever user interface.
- **Integrity Report System API**: provides a set of functions to manage flags.
- **RPMB API**: provides a set of functions to handle user data in RPMB
- **Thread support library API**: set of interfaces (functions, header files) for threaded programming commonly known as POSIX threads, or *Pthreads*. A single process can contain multiple threads, all of which are executing the same program. These threads share the same global memory (data and heap segments), but each thread has its own stack (automatic variables).
- **Math library API**: provides a set of math functions for floating point calculation.
- **Socket/Message API**: provides set of inter-process communications
- **Auxiliary API**: provides a set of API to provide POSIX-like interface. In Secure World, the feature coverage is limited to the minimum set of functions rather than fully supporting POSIX standard. Currently, only standard IO functions and string operations are supported by Auxiliary API.
- **NFC API**: provides a set of functions to manipulate device connected with NFC
- **SMC API**: provides a set of SMC command
- **Scrypto API**: provides a set of APIs supporting FIPS-certified crypto operation
- **Secure object API**: provides a set of APIs wrapping transient object with different access control and delegation
  - Some of TAs in Section 2.2.3 are using delegation among each other. For example, payment applications are delegated to use key created by keymanagement application. In another, biometric applications are delegated each other for handling biometric authentication information.
  - [SS ARC] has a guide how to check the validity and authenticity of delegated object.

TEE Communication Agent protocol and Shared memory between REE and TEE:

The communications between REE and TEE rely on a dedicated driver (TZDev) in charge of translating IOCTL commands received from the Normal World components into appropriate call to Secure World.

When data need to be exchanged, shared memory between REE and TEE is defined.

More details can be found in the [SS TDS] document.

3.2.3 TEE Security Functionality

The TEE security functionality in the end-user phase consists of:

- TEE instantiation through a secure initialization process using assets bound to the SoC, that ensures the authenticity and contributes to the integrity of the TEE code running in the device;
- Isolation of the TEE services, the TEE resources involved and all the Trusted Applications from the REE;
- Isolation between Trusted Applications and isolation of the TEE from Trusted Applications;
- Protected communication interface between CAs and TAs within the TEE, including communication endpoints in the TEE;
- Trusted storage of TA and TEE data and keys
  - Cryptographic structure ensuring confidentiality, integrity and binding to the TEE;
  - Monitoring action ensuring consistency and atomicity;
  - Violation action ensuring integrity and rollback attempts at-runtime.
- Random Number Generator (DRBG NIST SP800-90A);
- Cryptographic API (cf. Table 3-1, [SS CRY]):
  - Generation and derivation of keys and key pairs;
  - Signature/verification scheme;
  - Encryption/decryption.
- TA instantiation that ensures the authenticity and contributes to the integrity of the TA code;
- Monotonic TA instance time, Monotonic TA persistent time;
- Correct execution of TA services;
- TEE firmware and software integrity verification;
- Prevention of downgrade of TEE firmware and software, downgrade of TA code and binary;
- Irreversible JTAG configurations for debug access.
  - Access control based on cryptographic authentication in production mode

Table 3-1 List of the cryptographic APIs

Category	Algorithm identifier	Key length
AES	AES_ECB_NOPAD, AES_CBC_NOPAD, AES_CTR, AES_CTS, AES_CBC_MAC_NOPAD, AES_CBC_MAC_PKCS5, AES_CBC_MAC_ISO9797_M2, AES_CMAC, AES_CCM, AES_GCM, AES_ECB_ISO9797_M1, AES_ECB_ISO9797_M2, AES_CBC_ISO9797_M1, AES_CBC_ISO9797_M2, AES_ECB_PKCS5, AES_ECB_PKCS7, AES_CBC_PKCS5, AES_CBC_PKCS7, AES_ECB_NOPAD_HW, AES_CBC_NOPAD_HW, AES_CTR_HW, AES_CTS_HW	128, 192, 256
	AES_XTS	128, 256
DES	DES_ECB_NOPAD, DES_CBC_NOPAD	56[1]
DES3	DES3_ECB_NOPAD, DES3_CBC_NOPAD, DES3_CBC_MAC_NOPAD, DES3_CBC_MAC_PKCS5	112, 168[2]

[1] Simple DES uses keys of length 64. However, 8 bits are used for parity control and then only 56 bits are variable.

[2] Triple DES uses keys of length 128 and 192. However, for each byte, one bit is used for parity control. Only the variable number of bits is considered here.

RSA Sign/Verify	RSASSA_PKCS1_V1_5_SHA1, RSASSA_PKCS1_V1_5_SHA224, RSASSA_PKCS1_V1_5_SHA256, RSASSA_PKCS1_V1_5_SHA384, RSASSA_PKCS1_V1_5_SHA512, RSASSA_PKCS1_PSS_MGF1_SHA1, RSASSA_PKCS1_PSS_MGF1_SHA224, RSASSA_PKCS1_PSS_MGF1_SHA256, RSASSA_PKCS1_PSS_MGF1_SHA384, RSASSA_PKCS1_PSS_MGF1_SHA512	256-4096, mult of 128 bits
RSA Encryption	RSAES_PKCS1_V1_5, RSAES_PKCS1_OAEP_MGF1_SHA1, RSAES_PKCS1_OAEP_MGF1_SHA224, RSAES_PKCS1_OAEP_MGF1_SHA256, RSAES_PKCS1_OAEP_MGF1_SHA384, RSAES_PKCS1_OAEP_MGF1_SHA512, RSA_NOPAD (Encryption)	256-4096, mult of 128 bits
DSA	DSA_SHA1,	512-1024, mult of 64 bits
	DSA_SHA224	2048
	DSA_SHA256	2048, 3072
DH	DH_DERIVE_SHARED_SECRET	256-2048, mult of 8 bits
Hash	SHA1, SHA224, SHA256, SHA384, SHA512, SHA1_HW, SHA256_HW, SHA512_HW	-
HMAC	HMAC_SHA1	80-512, mult of 8 bits
	HMAC_SHA224	112-512, mult of 8 bits
	HMAC_SHA256	192-1024, mult of 8 bits
	HMAC_SHA384, HMAC_SHA512	256-1024, mult of 8 bits
	HMAC_SHA1_HW, HMAC_SHA256_HW	Up to 512
	HMAC_SHA512_HW	Up to 1024
ECDSA	ECDSA_SHA224	224
	ECDSA_SHA256	256
	ECDSA_SHA384	384
	ECDSA_SHA512	521
ECDH	ECDH_P224	224
	ECDH_P256	256
	ECDH_P384	384
	ECDH_P512	521
Curve 25519	ED25519, X25519	256
Others	Bignum arithmetics (about 40 items), KDF, Random Generation, Key Gen (HASH, HMAC, CTR DRBG)	-

Please note that some algorithms and keys lengths usage is forbidden or restricted, following [ANSSI-PG-083]. Appropriate configuration is defined in [SS OGD].

The TEE security functionalities define the logical boundary of the TOE. The interfaces of this boundary are the hardware external interfaces and the software external interfaces, introduced in sections 3.2.1 and 3.2.2 respectively.

The security functionalities provided by the Trusted Applications in Section 2.2.3 are out of the scope of the TOE.

3.2.4 TEE operation modes

On production devices, the TEE OS software only runs in one normal release mode. For development use, a debug mode can be activated enabling certain debug features such as backtrace, core dump, verbose logging and profiling. In addition, debug keys for TA developers can be issued in this mode. The debug mode is not activated on production devices.

TEE hardware (i.e., memory) can be configured with debug access. Debug functionality will be disabled on production devices, by blowing OTP. However, a cryptographic operation can be performed to get access to debug channel. This cryptographic access control will be given at every attempt to access per device.

3.3 Life Cycle

The following table lists the sites involved in the TEE development and production:

Developer / Manufacturer Company Name	Legal Address	Contact	TOE-related sites	Site audits / date
Samsung Electronics	(16677) 129, Samsung-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do, Korea	TEEgris (m.sec.teesdk@samsung.com)	Samsung Electronics Co., Ltd. - Mobile eXperience (MX) Division	TL 9000 (including ISO 9001): 200002540 TLR6, valid until 2023-07-12
Samsung Electronics	(18448) 1-1 Samsungjeonja-ro Hwaseong-si, Gyeonggi-do, Korea	TEEgris (m.sec.teesdk@samsung.com)	Samsung Electronics Co., Ltd. Samsung LSI Business	ISO 9001: Certificate No: FM 707390

The TEE life cycle involves two parties:

- SoC vendor: Samsung Electronics Co., Ltd, Device Solutions, System LSI Business
- Trusted OS vendor, Device developer/maker: Samsung Electronics Co., Ltd, Device eXperience, Mobile eXperience (MX) Business

The TEE life cycle is split in seven phases [SS ALC]:

- Phase 1 corresponds to the TEE firmware & hardware design.
- Phase 2 corresponds to SoC manufacturing.
- Phase 3 corresponds to the software (TEE + REE) design.
- Phase 4 corresponds to the TEE manufacturing.
- Phase 5 corresponds to the TEE integration into Final Device
- Phase 6 corresponds to the device manufacturing. In this phase, the TEE is initialized and personalized, before delivery.
- Phase 7 stands for the end-usage of the device (TEE operation).

The TEE delivery point is at the end of phase 6.

Table 3-2 Life Cycle

Phases	Actors
1: Firmware / Hardware design	SoC vendor is in charge of the firmware & hardware design. The TEE hardware designer: • Designs the TEE hardware as part of the entire SoC design. • Designs the ROM code, which includes the Secure Boot, the JTAG policy, the authentication policy to download images, and the bootloader. The TEE firmware designer: • Design the firmware code, which include hardware cryptography to perform above policies.
2: SoC hardware manufacturing	SoC vendor is in charge of the SoC hardware manufacturing. The SoC manufacturer: • Produces the TEE hardware as part of the SoC; • Deliver ATF, any firmware/software related to the SoC; • Blows the HW random Key (root-of-trust for TEE Trusted Storage) and the HW. random ID in EFUSE to enable per-device encryption/decryption.
3: Software design	Samsung Mobile (Korea) is in charge of the TEE and REE software design. The TEE software developer: • Develops the TEE software; • Generate the root-of-trust key pair used during Secure Boot to check authenticity of the bootloader when it is enabled. The product software developer: • Designs additional REE components, such as Linux TZ kernel driver and daemon to connect with the kernel driver, which will be linked with the TEE in phase 4; • Designs Trusted Applications that will be integrated in phase 4.
4: TEE software manufacturing	Samsung Mobile (Korea) is in charge of the software manufacturing. The product manufacturer: • Develops the TEE software; • Develops Trusted Application to be pre-loaded; • Develops/integrates additional REE software.
5: TEE integration into Final device	Samsung Mobile (Korea) are in charge of the integration The TEE integrator: • Integrates the TEE firmware on SoC/Final device; • Integrates the TEE software on SoC/Final device; • Integrates the pre-loaded TAs; • Verifies the TEE and TA functionality on the SoC platform.
6: Device manufacturing	Samsung Mobile (Korea) is in charge of the device manufacturing. The product manufacturer: • Carries out the device assembling; • Blows the root-of-trust for Secure Boot public key hash in EFUSE. This key hash is used to check integrity of the key used to verify the bootloader signature; • Disable debug facilities and interfaces • Initializes the TEE; • Performs any other operation on the device (including loading or installation of Trusted Applications) before delivery to the end-user.
7: End-usage phase	The end user gets a device ready for use. The TEE OS vendor is responsible for maintenance. The Trusted Applications manager is responsible for the loading, installation, and removal of Trusted Applications post-

	issuance.

4 Conformance Claim

4.1 Conformance Claim to CC

This Security Target claims conformance to Common Criteria version 3.1 revision 5 ([CC1], [CC2], [CC3] and [CEM]).

It also claims to be CC Part 2 and Part 3 extended.

CC Part 2 is extended with the security functional components “FCS_RNG.1 Random numbers generation” and “FPT_INI.1 TSF initialization”.

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

The TEE Time and Rollback and TEE Debug PP-Modules are CC Part 2 conformant.

4.2 Conformance Claim to a Package

This Security Target and associated PP-Configuration claims conformance to EAL 2 augmented with AVA_VAN_AP.3, as defined in [TEE PP TD 1.3].

4.3 Conformance Claim to a PP

4.3.1 PP-Configuration

This Security Target claims strict conformance to [TEE PP TD 1.3]. It includes the two defined PP-modules:

- TEE Time and Rollback
- TEE Debug

4.3.2 PP additions

Moreover, this Security Target integrates additional threats, assumptions, security objectives for the TOE and the environment and SFR to manage peripherals through the Trusted User Interface.

These additions are inspired by TUI PP-module document [TEE PP TUI 1.0], which is not a certified PP-module but provides a consistent security problem, objectives and SFRs.

Additions to the [TEE PP 1.3] are clearly identified with a visual color code:

- Violet identifies Debug PP-module from [TEE PP D 1.3]
- Blue identifies Time and Rollback PP-module from [TEE PP T 1.3]
- Green identifies additions for Trusted User Interface support
- Marron identifies a modification from an initial text (from base PP TEE or from PP-modules). It comes with an Application Note to justify the modification.

5 Security Problem Definition

This chapter is mainly extracted from PP TEE and PP-modules.

5.1 Assets

This section presents the assets of the TOE and their properties: authenticity, consistency, integrity, confidentiality, monotonicity, randomness, atomicity, read-only, and device binding (cf. section 1.3 for the definitions).

For details on modifications and additions by the PP-modules, please refer to [TEE PP TD 1.3].

TEE identification

TEE identification data that is globally unique among all TEEs whatever the manufacturer, vendor or integrator.

Properties: Unique and non-modifiable.

Application Note:

This data is stored as binary format of TEE software.

RNG

Random Number Generator.

Properties: Unpredictable random numbers, sufficient entropy.

TA code

The code of the installed Trusted Applications. This data is typically stored in external non-volatile memory which is shared with the REE and potentially accessible by it.

Properties: Authenticity, consistency, and integrity.

Application Note:

Integrity of TA code means that the value successfully read from a storage location is the last value that was written to this location.

TA data and keys

Data and keys managed and stored by TAs using the TEE security services. Data and keys are owned either by the user (the owner of the TEE-enabled device) or by the TA service provider. This data is typically stored in external non-volatile memory which is shared with the REE and potentially accessible by it.

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

Application Note:

Integrity of storage means that the value successfully read from a storage location is the last value that was written to this location.

TA instance time

Monotonic time available during TA instance lifetime. Not affected by transitions through low power states. Not persistent over TEE reset or TA shut-down.

Properties: Monotonicity.

TEE runtime data

TEE runtime data includes execution variables, runtime context, etc. This data is stored in volatile memory.

Properties: Consistency (or integrity as these notions are equivalent for non-persistent data) and confidentiality, including random numbers generated by the TEE.

TEE persistent data

TEE persistent data includes cryptographic keys (for instance keys to authenticate TA code) and TA properties. This data is typically stored in external non-volatile memory which is shared with REE and potentially accessible by it.

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

TEE firmware and software

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

Properties: Authenticity, integrity, confidentiality.

Application Note:

TEE binary is in fact composed of two elements: firmware and software. TEE firmware only refers to the part developed by SoC provider and TEE software refers to the part developed by Samsung MX section. Both have the same level of security.

Confidentiality is also required for TEEgris products. To keep secret TEE firmware and software participate to all assets security.

TEE initialisation code and data

Initialisation code and data (for instance cryptographic certificates) used from device power-on up to the complete activation of the TEE security services.

Properties: Integrity.

TEE storage root of trust

The root of trust of the TEE storage that is used to bind the stored data and keys to the TEE.

Properties: Integrity and confidentiality.

Application Note:

For TEEgris products, this root of trust is derived from cryptographic hash stored in the OTP memory of the SoC. Confidentiality is ensured by the fact that the asset remains inside the SoC.

TEE debug authentication key

The cryptographic key used to authenticate the TEE Debug Administrator for granting access to debug features.

Properties: Integrity, confidentiality and device binding.

Application Note:

TEEgris products also offer binding between the device and the TEE debug authentication key to enforce the confidentiality on the key and the authenticity of the Debug administrator.

TA persistent time

Monotonic TA time between two time setting operations performed by any instance of the TA and persistent over TEE reset.

Properties: Monotonicity.

TEE rollback detection data

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

Properties: Integrity.

(TA) EXCHANGED_DATA

This covers data transmitted between a locked peripheral and the calling TA. It represents runtime data.

For an input peripheral, the EXCHANGED_DATA represents INPUT_DATA captured by the input peripheral through its capture functionality on behalf of the calling TA.

For an output peripheral, the EXCHANGED_DATA represents OUTPUT_DATA displayed or presented via the output peripheral on behalf of the calling TA.

Properties (in the TOE): consistency, confidentiality.

Application Note:

The communication from the peripheral to the TEE is protected being physically non-accessible via TEE hardware.

PERIPHERAL_FIRMWARE

The peripheral's firmware. To a minimum, this includes the peripheral's driver. Such code is persistent and lies inside the TOE boundary.

Properties: integrity, authenticity, rollback protection.

Application Note:

This asset applies to peripherals that are under TEE control.

Time and Rollback PP-module being implemented, peripheral firmware benefits of integrity protection.

TUI_RUNTIME_DATA

TUI associated runtime data. Such data may include handles to events or to input runtime data.

Properties: consistency, confidentiality.

PERIPHERAL_SETUP

The peripheral's configuration data and settings. Such data is persistent.

Properties: integrity, authenticity.

Application Note:

This asset applies to all peripherals.

~~SECURITY_INDICATOR~~

~~The code and data of security indicator(s), which is a hardware output peripheral or a software mechanism (e.g. a dedicated display area) that is always under the control of the TEE and is not accessible to the REE or the TAs. The security indicator's state shows whether or not the associated peripheral is secured.~~

~~Security properties: Integrity.~~

Application Note:

Security Indicator option is not implemented.

5.2 Users

There are two kinds of users of the TOE: Trusted Applications, which use the TOE services through the TEE Internal Core API, and the Regular Execution Environment, which uses the services exported by the Trusted Applications.

Debug PP-module adds a third user who is the Debug Administrator.

Trusted Application (TA)

All Trusted Applications running on the TEE are users of the TOE, through the TEE Internal Core API.

Regular Execution Environment (REE)

The Regular Execution Environment, hosting the Regular OS, the TEE Client API, and the Client Applications that use the services of the Trusted Applications, is a user of the TOE.

TEE Debug Administrator

The TEE Debug Administrator who can be granted access to TEE debug features.

5.3 Threats

Attackers' profiles and threat models are deeply described in Annex A of PP TEE, and Threat chapters of PP TEE and TUI PP-Module document. Definitions are not be repeated here.

T.ABUSE_FUNCT

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

An attacker manages to instantiate an illegal TEE or to start up the TEE 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: TEE initialisation code and data (integrity), TEE runtime data (confidentiality, integrity), RNG (confidentiality, integrity), TA code (authenticity, consistency).

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

Application Note:

Attack paths may consist in, for instance, using commands in unexpected contexts or with unexpected parameters, impersonating authorized entities, or exploiting badly implemented reset functionality that provides undue privileges.

In particular, a fake application running in the Regular OS which masquerades as a security application running in the TEE can grab PINs and passwords and run the real security application on behalf of the user. However, such a threat cannot be countered by the TEE alone and must be taken into account in the design of the service, for instance by using an applicative authenticated communication channel between the client and the TA.

T.CLONE

An attacker manages to copy TEE related data from one device to a second device and makes this device accept them as genuine data.

Assets threatened directly: All data and keys (authenticity, device-binding), TEE identification data (authenticity, integrity).

T.FLASH_DUMP

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

Assets threatened directly (confidentiality, authenticity, consistency/integrity): TA data and keys, TEE persistent data.

Application Note:

An attack path consists for instance in performing a (partial) memory dump through the REE, purely via software or with a USB connection.

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

T.IMPERSONATION

An attacker impersonates a Trusted Application to gain unauthorized access to the services and data of another Trusted Application.

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

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

T.ROGUE_CODE_EXECUTION

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

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

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

Application Note:

Importation of code within the REE is out of control of the TEE.

T.PERTURBATION

An attacker modifies the behaviour of the TEE or the behaviour of a TA in order to disclose or modify sensitive data or to force the TEE or the TA to execute unauthorized services.

Assets threatened directly: TEE initialisation code and data (integrity), TEE storage root of trust

(confidentiality, integrity), TEE runtime data (confidentiality, integrity), RNG (confidentiality, integrity).

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

Application Note:

Unauthorized use of commands (one or many incorrect commands, undefined commands, hidden commands, invalid command sequence) or buffer overflow attacks (overwriting buffer content to modify execution contexts or gaining system privileges) are examples of attack paths. The TEE can also be attacked through REE or TA "programmer errors" that, for example, exploit multi-threading, context/session management, or closed sessions; or by triggering system resets during execution of commands by the TEE.

T.RAM

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

Assets threatened directly: TEE initialisation code and data (integrity), TEE storage root of trust (confidentiality, integrity), TEE runtime data (confidentiality, integrity), RNG (confidentiality, integrity).

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

Application Note:

When the REE and the TEE share some memory, an attack path consists in a (partial) memory dump (read/write) by the REE.

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

T.RNG

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

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

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

T.SPY

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

Assets threatened directly (confidentiality): All data and keys, TEE storage root of trust.

Application Note:

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

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

T.TEE_FIRMWARE_DOWNGRADE

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

Assets threatened directly: TEE firmware and software (integrity).

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

T.STORAGE_CORRUPTION

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

Assets threatened directly: TEE storage root of trust (confidentiality, integrity), TEE persistent data (confidentiality, consistency/integrity), TEE firmware and software (authenticity, integrity), TA data and keys (confidentiality, authenticity, consistency/integrity), TA instance time (integrity), TA code (authenticity, consistency/integrity).

Application Note:

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

T.ABUSE_DEBUG

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

Assets threatened directly: TEE initialisation code and data (integrity), TEE runtime data (confidentiality, integrity), RNG (confidentiality, integrity), TA code (authenticity, consistency/integrity).

Assets threatened indirectly: TA data and keys (confidentiality, authenticity, consistency/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 TEE debug mode.

T.ROLLBACK

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

Assets threatened directly (confidentiality, integrity): TA data and keys, TEE persistent data, TA code.

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

Application Note:

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

T.TA_PERSISTENT_TIME_ROLLBACK

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

Assets threatened directly: TA persistent time (integrity).

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

Application Note:

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

T.SHARED_FUNCTION_ACCESS

An attacker intercepts and/or modifies exchanged data, taking advantage of a shared access to the peripheral's data-handling functionality and thus to the runtime data exchanged between the peripheral and the TOE.

Assets threatened indirectly: EXCHANGED_DATA (sent through the communication channel between the peripheral and the TOE).

Application Note:

For input peripherals, the goal of such an attack can be manifold, for instance:

- To extract genuine captured input data for subsequent replay;
- To alter genuine captured data before they are transmitted to the TOE.

T.EXTRACT_EXCHANGED_DATA

An attacker intercepts and extracts the data sent by/to the peripheral to/by the TUI through a communication channel.

Assets threatened indirectly: EXCHANGED_DATA (sent through the communication channel between the peripheral and the TUI).

T.MODIFY_EXCHANGED_DATA

An attacker intercepts and modifies the data sent by/to the peripheral to/by the TUI through a communication channel.

Assets threatened indirectly: EXCHANGED_DATA (sent through the communication channel between the peripheral and the TUI).

T.INJECT_EXCHANGED_DATA

An attacker injects data through the communication channel between the peripheral and the TUI.

Assets threatened indirectly: EXCHANGED_DATA (sent through the communication channel between the peripheral and the TUI).

Application Note:

For input peripherals, extraction and injection of the input data may be the two steps of a replay attack. First, an attacker intercepts and steals the captured input data. The interception is done when the input peripheral acquires input data from a genuine user and then sends it to the TOE through a communication channel. As a second step, the attacker injects or replays the stolen input data. The goal of this attack is to bypass the input peripheral.

T.MODIFY_FIRMWARE

An attacker modifies the firmware (drivers) pertaining to the locked peripheral to alter the behavior of the TUI.

Assets threatened directly: PERIPHERAL_FIRMWARE.

T.INJECT_DATA

An attacker bypasses the TUI and the peripheral and injects (malicious or corrupt) runtime data.

Assets threatened directly: EXCHANGED_DATA.

T.EXTRACT_DATA

An attacker extracts data outside the TOE by intercepting the data transmitted between the TUI and the TA that locked the peripheral.

Assets threatened directly: EXCHANGED_DATA

T.MODIFY_DATA

An attacker modifies the data transmitted between the TUI and the TA that locked the peripheral.

Assets threatened directly: EXCHANGED_DATA.

T.RESIDUAL

An attacker extracts unprotected residual security-relevant data during a TUI's session or from the cache.

This attack covers multiple scenarios:

- The attacker takes advantage of a flaw in the user interface of the TOE and gets access to the memory content, the cache or relevant temporary data;
- The attacker takes advantage of residual information such as residual runtime data at the level of the peripheral.

Assets threatened directly: EXCHANGED_DATA, TUI_RUNTIME_DATA, PERIPHERAL_SETUP.

T.CORRUPT_RUNTIME_DATA

An attacker corrupts runtime data, such as a handle to runtime data, a handle to the locked peripheral or a reference provided to the TA. Such data can be manipulated to alter the system's expected behavior.

Assets threatened directly: TUI_RUNTIME_DATA.

Application Note:

In a GlobalPlatform compliant implementation, this threat stands, for instance, for overwriting the EventSourceHandle.

T.CORRUPT_SETUP

An attacker modifies the setup of the locked peripheral.

Assets threatened directly: PERIPHERAL_SETUP.

T.PERIPHERAL_ACCESS

An attacker accesses (reads or writes) the peripheral while another TA has locked it.

Assets threatened directly: EXCHANGED_DATA.

Application Note:

This may be, for example, by using an undocumented feature of the peripheral that allows dumping its memory to some location.

5.4 Organisational Security Policies

OSP.INTEGRATION_CONFIGURATION

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

Application Note:

Integration and configuration of the TEE is defined in [SS OGD].

OSP.SECRETS

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

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

5.5 Assumptions

A.PROTECTION_AFTER_DELIVERY

It is assumed that the TOE and its assets are protected by the operational environment after delivery and before entering the end-usage phase. It is assumed that the persons using the TOE in the operational environment have the required skills to understand and apply the security guidelines.

Application Note:

TEEgris 5.0 is delivered after Phase 6 and this assumption applies only for Phase 7 - End usage.

Note that the operational environment is out of scope of the evaluation.

A.TA_MANAGEMENT

A well-defined TA identification and TA signature policy exist which ensures the authenticity of the application and prevents impersonation. The entity responsible for TA identification and TA signature ensures that these operations are performed in a controlled environment through dedicated procedures, which prevent, by technical and/or organizational means from:

- Assigning the same identification to different applications;
- Signing applications that have not been identified following the applicable procedures;
- Accessing to signature keys without authorization.

TA developers carefully consider the following TEE principles with regard to TA and Trusted Storage management during the development of their applications:

- The TA identification (or TA identity) is composed of a TA UUID and a TA Authority ID (if defined) and is managed by the entity in charge of signing the application;

- The TEE does not provide TA install/uninstall functions;
- TA loading and TA session opening are performed at the same time upon successful verification of the TA code signature, provided no “single-instance” application with the same TA identification is already running;
- Multiple application versions with the same TA identity may run concurrently and access the same set of data provided the TA is “multi-instance”;
- The ownership of persistent data stored in a Trusted Storage object associated with a given TA identity is automatically granted to any application instance that is loaded with such TA identity;
- Trusted Storage objects are never erased by the TEE (no TA install/uninstall functionality provided) and then remain accessible without any limitation of time or kind of operation (e.g. creation, read, write, delete).

Consequently, TA developers are assumed to internalize the management of TA life-cycle and of TA persistent data life-cycle within the TA itself.

Application Note:

This assumption has been modified to describe the TA management policy according to the Application Note in the Protection Profile. It also provides precisions of TA developers role and TEE features in TA management.

Rationale: The assumption already took into account the presence of a TA management policy. The modifications above only explain to TA developers how to develop their TA according to this TA management policy. Therefore, this modification doesn’t counter any threat (even partially) and doesn’t enforce any OSP (even partially).

A.TA_DEVELOPMENT

TA developers are assumed to comply with the TA development guidelines set by the TEE provider. In particular, TA developers are assumed to consider the following principles during the development of the Trusted Applications:

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

Application Note:

Recommendations for the development of secure TAs is provided in [SS OGD].

~~A.ROLLBACK~~

~~It is assumed that TA developers do not rely on protection of TA data and keys as trusted storage against full rollback.~~

Application Note:

This assumption is discarded, according to TEE Time and Rollback PP-module requirement. The TOE enforces the anti-rollback protection.

A.TA_DEVELOPMENT_TUI

TA developers shall follow the guidance documentation provided with the TOE and set by the TEE

provider. This documentation shall include the constraints that need to be respected by the applications for each peripheral to ensure that data can be interpreted.

Application Note:

This assumption is meant to ensure that data for which integrity shall be ensured by the TUI matches the range of the peripheral used by the application. For example, the microphone may not be capable of capturing all audible sounds. The application developer shall ensure that the application fits within these limits.

Rationale: This assumption complements A.TA_DEVELOPMENT. It doesn't counter any threat (even partially) and doesn't enforce any OSP (even partially) from PP-Base, Debug PP-module and Time and Rollback PP-module because it only concerns additional guidance on peripheral management.

A.NO_RESIDUAL_DATA

The function of the peripheral that handles data exchanged between the peripheral and the TOE does not store residual exchanged data. The exchanged data is deleted between any two consecutive operations.

Application Note:

Rationale: This assumption only relies on TUI assets introduced in TUI PP-module. It doesn't counter any threat (even partially) and doesn't enforce any OSP (even partially) from PP-Base, Debug PP-module and Time and Rollback PP-module because the TUI assets were not part of them.

6 Security Objectives

This chapter is mainly extracted from PP TEE and PP-modules.

6.1 Security Objectives for the TOE

O.CA_TA_IDENTIFICATION

The TEE shall provide means to protect the identity of each Trusted Application from use by another resident Trusted Application and to distinguish Client Applications from Trusted Applications.

Application Note:

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

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

O.KEYS_USAGE

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

O.TEE_ID

The TEE shall ensure statistical uniqueness of the TEE identifier when generated by the TEE. It shall also ensure that it is non-modifiable and provide means to retrieve this identifier.

Application Note:

TEE identifier is generated during TEE initialization. It is computed from Chip ID (which is unique by Chip). Then it is included in the binary code, signed, and stored on TEE with binary.

O.INITIALISATION

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

- The integrity of the TEE initialisation code and data used to load the TEE firmware and software;
- The authenticity of the TEE firmware and software;
- The binding of the TEE firmware and software to the SoC of the device;
- The protection against TEE firmware and software downgrade attacks.

O.INSTANCE_TIME

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

O.OPERATION

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

- Protect itself against abnormal situations caused by programmer errors or violation of good practices by the REE (and the CAs indirectly) or by the TAs;
- Control access to its services by the REE and TAs: The TEE shall check the validity of any operation requested from either the REE or a TA, at any entry point into the TEE;
- 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 multi-threading or context/session management) might become attack-enablers. However, the TEE must guarantee the stability and security of its resources and services independent of the REE, which may have been corrupted. In any case, a Trusted Application must not be able to use a programmer error on purpose to circumvent the TEE security functionality.
- Software in the REE must not be able to call directly to TEE resources or functions. The REE software must go through protocols such that the Trusted OS or Trusted Application performs the verification of the acceptability of the operation that the REE software has requested.

O.RNG

The TEE 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.

O.RUNTIME_CONFIDENTIALITY

The TEE shall ensure that confidential TEE runtime data and TA data and keys are protected against unauthorized disclosure. In particular:

- The TEE shall not export any sensitive data, random numbers or secret keys to the REE;
- The TEE shall grant access to sensitive data, random numbers or secret keys only to authorized TAs;
- The TEE shall clean up sensitive resources as soon as it can determine that their values are no longer needed.

O.RUNTIME_INTEGRITY

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

O.TA_AUTHENTICITY

The TEE shall verify the authenticity of the Trusted Applications' binary code.

Application Note:

Verification of authenticity of TA code can be performed together with verification of the TEE firmware and software if both are bundled together or during the loading of the TA code in volatile memory.

O.TA_ISOLATION

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

Application Note:

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

O.TEE_DATA_PROTECTION

The TEE shall ensure the authenticity, consistency, and confidentiality of TEE persistent data.

O.TEE_ISOLATION

The TEE shall prevent the REE and the TAs from accessing the TEE's own execution and storage space and resources.

Application Note:

This objective contributes to the enforcement of the correct execution of the TEE. Note that resource allocation can change during runtime as long as it does not break isolation between resources used by the TEE and the REE/TAs.

O.TRUSTED_STORAGE

The TEE shall provide Trusted Storage services for persistent TA data and keys such that the following properties are enforced: confidentiality, authenticity, and consistency.

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

The Trusted Storage shall be bound to the host device, which means that the stored data cannot be read in another device.

O.DEBUG

The TEE shall authenticate the TEE Debug Administrator before granting access to the TEE debug functionality.

O.ROLLBACK_PROTECTION

The TEE shall prevent unauthorized rollback by:

- Monitoring integrity of TEE persistent data, TA data or keys, or TA code;
- Reacting to possible integrity violation so that the security is always enforced.

Application Note:

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

O.TA_PERSISTENT_TIME

The TEE shall provide TA persistent time, which is persistent over TEE reset. The TEE shall ensure

that:

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

O.PERIPHERAL_INITIALISATION

The TOE shall ensure that the peripheral is started through a secure initialization process that ensures the integrity of the peripheral’s initialization code and data, and the authenticity of the peripheral firmware.

The TOE shall ensure that all peripheral code and data are bound to the SoC of the device.

Application Note:

This objective is the extension of the objective O.INITIALIZATION. It is included here to highlight the fact that the peripheral is indeed integral to the TEE.

Application Note:

Initialisation process of the peripheral happens when TUI API is called by TEE. Peripheral code and data integrity and authenticity is checked at this moment. For input peripherals, this excludes the code of the capture function, which is out of the scope of the TOE. For output peripherals, this excludes the output presentation function which is out of the scope of the TOE.

O.PROTECTED_COMMUNICATION_CHANNEL

The TOE shall provide the necessary means for protecting the communication channel between the peripheral and the TUI, i.e. it will isolate and protect it from unauthorized access by the REE or other TAs, which could lead to modification, injection or disclosure of the exchanged data.

Application Note:

This means that the TOE provides appropriate access control to the communication channel that carries the exchanged data.

O.PREVENT_RESIDUAL_DATA

The TOE shall ensure that when a locked peripheral is released, all residual data under TOE control captured or presented through this peripheral are erased at the level of this peripheral. Data must be deleted or invalidated between any two consecutive operations of the peripheral.

O.DATA_ACCESS

The TOE shall ensure that, when a peripheral is locked by a TA, only that TA can access (read or write) the data exchanged with the peripheral.

O.FUNCTION_ACCESS

The TOE shall ensure that:

- the data exchanged between the TOE and the peripheral is handled by the expected wired peripheral and that
- at the time of the usage, the calling TA has exclusive access to the peripheral’s function handling the exchanged (input/output) data, and therefore to the exchanged data.

O.SAFE_RELEASE

The TOE shall ensure that only the TA that locked a peripheral or the TOE itself or an external event (e.g. power event) can initiate a release of that peripheral.

6.2 Security Objectives for the Operational Environment

OE.INTEGRATION_CONFIGURATION

Integration and configuration of the TEE by the device manufacturer shall comply with the security guidelines defined by the TEE provider, which must include all recommendations issued from the TEE evaluation.

OE.SECRETS

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

OE.PROTECTION_AFTER_DELIVERY

The TEE and its assets shall be protected after delivery and before entering the end-usage phase. The personnel using the TEE in the operational environment shall have the required skills to understand and apply the security guidelines.

Application Note:

TEEgris 5.0 is delivered after Phase 6 and this assumption applies only for Phase 7 - End usage.

OE.TA_MANAGEMENT

The entity responsible for TA identification and TA signature shall ensure that these operations are performed in a controlled environment through dedicated procedures, which prevent, by technical and/or organisational means from:

- Assigning the same identification to different applications;
- Signing applications that have not been identified following the applicable procedures;
- Unauthorized access to signature keys.

TA developers carefully must consider the following TEE principles with regard to TA and Trusted Storage management during the development of their applications:

- The TA identification (or TA identity) is composed of a TA UUID and a TA Authority ID (if defined) and is managed by the entity in charge of signing the application;
- The TEE does not provide TA install/uninstall functions;
- TA loading and TA session opening are performed at the same time upon successful verification of the TA code signature, provided no “single-instance” application with the same TA identification is already running;
- Multiple application versions with the same TA identity may run concurrently and access the same set of data provided the TA is “multi-instance”;
- The ownership of persistent data stored in a Trusted Storage object associated with a given TA identity is automatically granted to any application instance that is loaded with such TA identity;
- Trusted Storage objects are never erased by the TEE (no TA install/uninstall functionality provided) and then remain accessible without any limitation of time or kind of operation

(e.g. creation, read, write, delete).

Consequently, TA developers must internalize the management of TA life-cycle and of TA persistent data life-cycle within the TA itself.

Application Note:

This applies to all the phases of the TEE life cycle provided it allows TA management, before and after TOE delivery point.

This Objective for the Security Environment has been modified in accordance with the modifications performed on the assumption A.TA_MANAGEMENT. It provides information and instructions to TA developer how to develop their TA in accordance with the TA management policy.

OE.TA_DEVELOPMENT

TA developers shall comply with the TA development guidelines set by the TEE provider. In particular, TA developers shall apply the following security recommendations during the development of Trusted Applications:

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

Application Note:

This includes the security guidelines that fulfil AGD_OPE.1 and contain all the recommendations for the development of secure TAs.

~~OE.ROLLBACK~~

~~TA developers shall not rely on the protection of TEE persistent data, TA data and keys, and TA code against rollback between two reset operations.~~

Application Note:

This Security Objective for the Environment is discarded, regarding that Time and Rollback PP-module is used and anti-rollback protection is enforced by the TOE (cf. O.ROLLBACK_PROTECTION).

~~OE.DISABLED_DEBUG~~

~~All TEE debug interfaces shall be disabled in the end-user phase.~~

Application Note:

This Security Objective for the Environment is discarded, regarding that Debug PP-Module is used and Debug interface protection is enforced by the TOE (cf. O.DEBUG).

OE.NO_RESIDUAL_DATA

The operational environment ensures that no residual exchanged (input/output) data is stored at the level of the peripheral. The exchanged data is deleted between any two consecutive capture or presentation operations.

OE.TA_DEVELOPMENT_TUI

TA developers shall comply with the guidance documentation provided with the TOE. In particular, for each peripheral, TA developers shall respect the constraints necessary for ensuring that data can be interpreted. For each peripheral the TA developers shall consider the characteristics specified in the guidance documentation. In particular, they consider:

- the peripheral's type, i.e. input, output, or I/O peripheral;
- whether the peripheral supports exclusive access;
- whether it relies on a Security Indicator and how the information indicated by it is to be interpreted.

6.3 Security Objectives rationale

	O.CA_TA_IDENTIFICATION	O.KEYS_USAGE	O.TEE_ID	O.INITIALISATION	O.INSTANCE_TIME	O.OPERATION	O.RNG	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TEE_ISOLATION	O.TRUSTED_STORAGE	O.DEBUG	O.ROLLBACK_PROTECTION	O.TA_PERSISTENT_TIME	O.PERIPHERAL_INITIALISATION	O.PROTECTED_COMMUNICATION_CHANNEL	O.PREVENT_RESIDUAL_DATA	O.DATA_ACCESS	O.FUNCTION_ACCESS	O.SAFE_RELEASE	OE.INTEGRATION_CONFIGURATION	OE.SECRETS	OE.PROTECTION_AFTER_DELIVERY	OE.TA_MANAGEMENT	OE.TA_DEVELOPMENT	OE.NO_RESIDUAL_DATA	OE.TA_DEVELOPMENT_TUI
T.ABUSE_FUNCT		x		x		x		x	x	x		x	x															x		
T.CLONE			x	x				x	x			x		x																
T.FLASH_DUMP														x																
T.IMPERSONATION	x					x			x																					
T.ROGUE_CODE_ EXECUTION				x		x		x	x	x		x		x										x		x				
T.PERTURBATION				x	x	x		x	x	x	x	x	x				x													
T.RAM				x				x	x		x		x																	
T.RNG				x			x	x	x																					
T.SPY								x			x		x	x																	
T.TEE_FIRMWARE_ DOWNGRADE				x																				x		x				
T.STORAGE_CORRUPTION				x		x				x		x		x		x														
T.ABUSE_DEBUG															x															
T.ROLLBACK																x														

	O.CA_TA_IDENTIFICATION	O.KEYS_USAGE	O.TEE_ID	O.INITIALISATION	O.INSTANCE_TIME	O.OPERATION	O.RNG	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TEE_ISOLATION	O.TRUSTED_STORAGE	O.DEBUG	O.ROLLBACK_PROTECTION	O.TA_PERSISTENT_TIME	O.PERIPHERAL_INITIALISATION	O.PROTECTED_COMMUNICATION_CHANNEL	O.PREVENT_RESIDUAL_DATA	O.DATA_ACCESS	O.FUNCTION_ACCESS	O.SAFE_RELEASE	OE.INTEGRATION_CONFIGURATION	OE.SECRETS	OE.PROTECTION_AFTER_DELIVERY	OE.TA_MANAGEMENT	OE.TA_DEVELOPMENT	OE.NO_RESIDUAL_DATA	OE.TA_DEVELOPMENT_TUI
T.TA_PERSISTENT_TIME_ROLLBACK																	x													
T.SHARED_FUNCTION_ACCESS																			x	x		x								
T.EXTRACT_EXCHANGED_DATA																			x											
T.MODIFY_EXCHANGED_DATA																			x											
T.INJECT_EXCHANGED_DATA																			x											
T.MODIFY_FIRMWARE						x												x												
T.INJECT_DATA									x																					
T.EXTRACT_DATA								x																						
T.MODIFY_DATA									x																					
T.RESIDUAL								x																						
T.CORRUPT_RUNTIME_DATA									x																					
T.CORRUPT_SETUP									x			x																		
T.PERIPHERAL_ACCESS																					x		x							
OSP.INTEGRATION_CONFIGURATION																								x						
OSP.SECRETS																									x					
A.PROTECTION_AFTER_DELIVERY																										x				
A.TA_MANAGEMENT																											x			
A.TA_DEVELOPMENT																												x		
A.TA_DEVELOPMENT_TUI																														x

	O.CA_TA_IDENTIFICATION	O.KEYS_USAGE	O.TEE_ID	O.INITIALISATION	O.INSTANCE_TIME	O.OPERATION	O.RNG	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TEE_ISOLATION	O.TRUSTED_STORAGE	O.DEBUG	O.ROLLBACK_PROTECTION	O.TA_PERSISTENT_TIME	O.PERIPHERAL_INITIALISATION	O.PROTECTED_COMMUNICATION_CHANNEL	O.PREVENT_RESIDUAL_DATA	O.DATA_ACCESS	O.FUNCTION_ACCESS	O.SAFE_RELEASE	OE.INTEGRATION_CONFIGURATION	OE.SECRETS	OE.PROTECTION_AFTER_DELIVERY	OE.TA_MANAGEMENT	OE.TA_DEVELOPMENT	OE.NO_RESIDUAL_DATA	OE.TA_DEVELOPMENT_TUI
A.NO_RESIDUAL_DATA																												x	

T.ABUSE_FUNCT

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

- O.INITIALISATION ensures that the TEE 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.TA_AUTHENTICITY ensures that the authenticity of TA code is verified
- O.TEE_DATA_PROTECTION ensures that the data used by the TEE are authentic and consistent
- O.TEE_ISOLATION enforces the separation between the TEE and the outside (REE and TAs)
- O.KEYS_USAGE controls the usage of cryptographic keys
- OE.TA_DEVELOPMENT enforces TA 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.TEE_ID provides the unique TEE identification means
- O.INITIALISATION ensures that the TEE is bound to the SoC of the device
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data, particularly TSF data used to bind the TEE to the device
- O.RUNTIME_INTEGRITY prevents unauthorized modification at runtime of security functionalities or data used to detect or prevent cloning
- O.TEE_DATA_PROTECTION prevents the TEE from using TEE data that is inconsistent or not authentic
- O.TRUSTED_STORAGE ensures that the trusted storage is bound to the device and prevents the TEE from using data that is inconsistent or not authentic

- ~~O.RNG ensures that the TEE identifier is in fact unique when generated inside the TOE~~

Application Note:

The TEE identifier is not generated by a random.

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_TA_IDENTIFICATION ensures the protection of Client identities and the possibility to distinguish Client Applications and Trusted Applications
- 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.INITIALISATION ensures that the TEE security functionality is correctly initialized and the integrity of TEE firmware and software
- O.OPERATION ensures correct operation of the security functionality
- O.RUNTIME_CONFIDENTIALITY covers runtime TEE data which might influence the behaviour of the TEE
- O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified
- O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime
- O.TEE_DATA_PROTECTION covers persistent TEE data which might influence the behaviour of the TEE
- 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 after delivery and before the end-user phase.

T.PERTURBATION

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

- O.INITIALISATION ensures that the TEE 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 proper management of failures
- O.RUNTIME_CONFIDENTIALITY covers runtime TEE data which might influence the behaviour of the TEE
- O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified
- O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime

- O.TA_ISOLATION ensures the separation of the TA
- O.TEE_DATA_PROTECTION covers persistent TEE data which might influence the behaviour of the TEE
- O.TEE_ISOLATION enforces the separation between the TEE and the outside (REE and TAs).
- O.TA_PERSISTENT_TIME ensures the reliability of persistent time stamps

T.RAM

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

- O.INITIALISATION ensures that the TEE security functionality is correctly initialized and that the initialisation process is protected from the REE
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data at runtime
- O.RUNTIME_INTEGRITY protects against unauthorized modification of code and data at runtime
- O.TA_ISOLATION provides a memory barrier between TAs
- O.TEE_ISOLATION provides a memory barrier between the TEE and the REE.

T.RNG

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

- O.INITIALISATION ensures the correct initialisation of the TEE security functions, particularly 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.TA_ISOLATION ensures the separation between TAs
- O.TEE_ISOLATION ensures that neither REE nor TAs can access TEE data
- O.TRUSTED_STORAGE ensures that data stored in the trusted storage locations is accessible by the TA owner only.

T.TEE_FIRMWARE_DOWNGRADE

The combination of the following objectives ensures protection against TEE firmware and software downgrade:

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

T.STORAGE_CORRUPTION

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

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

storage

- O.TEE_DATA_PROTECTION ensures that stored TEE data are genuine and consistent
- O.TRUSTED_STORAGE enforces detection of corruption of the TA's storage
- O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified
- O.INITIALISATION ensures that the firmware that is executed is the intended version
- O.ROLLBACK_PROTECTION ensures that TA cannot use rollbacked data like TEE data, TA code and TA data and keys

T.ABUSE_DEBUG

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

T.ROLLBACK

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

T.TA_PERSISTENT_TIME_ROLLBACK

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

T.SHARED_FUNCTION_ACCESS is covered by the following security objectives:

- O.FUNCTION_ACCESS ensures that the TA has exclusive access to the peripheral's function handling the exchanged (input/output) data, and therefore to the exchanged data.
- O.PROTECTED_COMMUNICATION_CHANNEL ensures that exchanged data cannot be intercepted or modified when transmitted to/by the peripheral from/to the TUI.
- O.PREVENT_RESIDUAL_DATA ensures that no residual data can be extracted subsequently at the level of the peripheral or from memory.

Application Note:

In [TEE PP TUI 1.0], this threat is countered by O.CAPTURE in the rationale. However, O.CAPTURE is not defined (it should be a typo). O.FUNCTION_ACCESS is appropriate to counter this treat and then a rationale has been added above.

T.MODIFY_EXCHANGED_DATA, T.INJECT_EXCHANGED_DATA, T.EXTRACT_EXCHANGED_DATA are covered by the following security objective:

- O.PROTECTED_COMMUNICATION_CHANNEL ensures that exchanged data cannot be intercepted, injected, or modified when transmitted to/by the peripheral from/to the TUI.

T.MODIFY_DATA, T.INJECT_DATA, T.CORRUPT_RUNTIME_DATA~~, and T.CORRUPT_SETUP~~ depend on runtime integrity. They are completely covered by the following security objective from Base-PP:

- O.RUNTIME_INTEGRITY ensures runtime integrity and prevents unauthorized modification of runtime data.

T.CORRUPT_SETUP depends on TEE data protection. It is completely covered by:

- O.RUNTIME_INTEGRITY ensures runtime integrity and prevents unauthorized modification of runtime data (including setup data).
- O.TEE_DATA_PROTECTION ensures that stored TEE data (including setup data) are genuine and consistent.

Application Note:

In the [TEE PP TUI 1.0], T.CORRUPT_SETUP is countered only by O.RUNTIME_INTEGRITY. However, setup data are identified as persistent data and then the threat is countered by the Security Objective for the TOE

O.TEE_DATA_PROTECTION as explained above.

T.EXTRACT_DATA and **T.RESIDUAL** depend on runtime confidentiality. They are completely covered by the following security objective defined in the [TEE PP 1.3]:

- O.RUNTIME_CONFIDENTIALITY ensures runtime confidentiality and prevents exposure or extraction of data.

T.MODIFY_FIRMWARE is covered by the following security objectives:

- O.PERIPHERAL_INITIALISATION ensures the integrity of the peripheral initialisation code and data, as well as the authenticity of the peripheral firmware.
- O.OPERATION defined in the [TEE PP] ensures correct operation of the security functionality.

T.PERIPHERAL_ACCESS is completely covered by the following security objectives:

- O.DATA_ACCESS ensures that when a peripheral is locked by a TA, only that TA can access the data exchanged with the peripheral.
- O.SAFE_RELEASE ensures that the peripheral is safely released either by the TA that locked it, by the TOE itself, or by an external event (e.g. power event).

OSP.INTEGRATION_CONFIGURATION

The objective OE.INTEGRATION_CONFIGURATION directly covers this OSP.

OSP.SECRETS

The objective OE.SECRETS directly covers this OSP.

A.PROTECTION_AFTER_DELIVERY

The objective OE.PROTECTION_AFTER_DELIVERY directly covers this assumption.

A.TA_MANAGEMENT

The objective OE.TA_MANAGEMENT directly covers this assumption.

Application Note:

Both A.TA_MANAGEMENT and OE.TA_MANAGEMENT have been modified from PP TEE but in the same way so the objective still directly and completely covers the assumption.

A.TA_DEVELOPMENT

The objective OE.TA_DEVELOPMENT directly covers this assumption.

A.TA_DEVELOPMENT_TUI

The objective OE.TA_DEVELOPMENT_TUI directly covers the assumption.

A.NO_RESIDUAL_DATA

The objective OE.NO_RESIDUAL_DATA directly covers the assumption.

7 Extended Requirements

7.1 FCS_RNG – Generation of Random Numbers

Family behaviour

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

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

Component levelling

There is only one level in this family.

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

Management: FCS_RNG.1

No management activities are foreseen.

Audit: FCS_RNG.1

No actions are defined to be auditable.

FCS_RNG.1 Random numbers generation

Hierarchical to: No other components.

Dependencies: No dependencies

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

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

7.2 FPT_INI - TSF Initialisation

Family behaviour

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

Component levelling

There is only one level in this family.

FPT_INI.1 Requires the provision of an initialisation function that brings the TSF into a secure operational state at power-on.

Management: FPT_INI.1

No management activities are foreseen.

Audit: FPT_INI.1

No actions are defined to be auditable.

FPT_INI.1 TSF initialisation

Hierarchical to: No other components.

Dependencies: No dependencies.

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

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

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

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

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

7.3 AVA_VAN_AP Vulnerability Analysis

Objectives

Vulnerability analysis is an assessment to determine whether potential vulnerabilities identified in the TOE could allow attackers to violate the SFRs and thus to perform 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 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.

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

Component levelling

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

AVA_VAN_AP.3 TEE 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, 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.

8 Security Requirements

8.1 Conventions

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

Operations on SFRs:

- Iterations are identified by “/xxx” where xxx is the identifier for the iterated SFR;
- Assignments and selections are in **bold**. These operations can be reported from the PP or can be instantiations of this Security Target;
- Refinements are in maroon, they can be strikethrough, if necessary;

8.2 Security Policies

Users stand for entities outside the TOE:

- Client Applications (CA), with security attribute "CA_identity" (CA identifier)
- Trusted Applications (TA), with security attribute "TA_identity" (TA identifier), "TA_properties".

Subjects stand for active entities inside the TOE:

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

Objects stand for passive entities inside the TOE:

- OB.TA_STORAGE (user data): Trusted Storage space of a TA, with security attributes "owner" (TA identifier), "inExtMem" (True/False) and "TEE_identity" (TEE identifier).
- OB.SRT (TSF data): The TEE Storage Root of Trust, with security attribute "TEE_identity" (TEE identifier).

Cryptographic objects are a special kind of TEE object:

- OB.TA_KEY (user data): (handle to a) user key (persistent or transient), with security attributes "usage", "owner" (TA 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 TA or to the TEE 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 and persistent TEE data that is 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 TA_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 TA_INSTANCE:
- OP.LOAD: Any operation used to get back persistent objects (data and keys) to be used by the TA
- 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 TEE on behalf of a TA_INSTANCE.

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

Runtime Data Information Flow Control SFP:

- Purpose: To control the flow of runtime data from and to executable entities and memory. This policy contributes to ensuring the integrity and confidentiality of runtime data
- Subjects: S.TA_INSTANCE, S.TA_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.

TA Keys Access Control SFP:

- Purpose: To control access to TA keys, which is granted to the TA that owns the key only. This policy contributes to the confidentiality of TA keys.
- Subjects: S.API, S.TA_INSTANCE and any other subject in the TEE
- Objects: OB.TA_KEY
- Security attributes: OB.TA_KEY.usage, OB.TA_KEY.owner, OB.TA_KEY.isExtractable, and S.API.caller
- Operations: OP.USE_KEY, OP.EXTRACT_KEY
- SFR instances: FDP_ACC.1/TA_Keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys, FMT_SMF.1.

Trusted Storage Access Control SFP:

- Purpose: To control access to TA storage where persistent TA data and keys are stored, which is granted on behalf of the owner TA only. This policy also enforces the binding of TA trusted storage to the TEE storage root of trust OB.SRT
- Subjects: S.API
- Objects: OB.TA_STORAGE, OB.SRT
- Security attributes: S.API.caller, OB.TA_STORAGE.owner, OB.TA_STORAGE.inExtMem, OB.TA_STORAGE.TEE_identity, and OB.SRT.TEE_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.

8.3 Security Functional Requirements

8.3.1 TEE Base-PP

8.3.1.1 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, TA_identity, TA_properties, privilege_agent_identity.**

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 TA;
- TA_identity: the availability of this attribute is that of the availability of the TA to clients, limited further by the TAs presence in the system;
- TA_properties: the lifetime of this attribute is that of the availability of the TA to clients, limited further by the TAs presence in the system;
- **privilege_agent_identity: the lifetime of this attribute is that of the lifetime of a SMC (call to the TEE not arising from a CA request).**

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, Trusted Application and the Linux Kernel Module (REE low level interface with the TEE). Note that Client Application requests are transferred to the TEE via a dedicated Linux driver (TzDev for TrustZone Driver) with root privileges.

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 TA) identity is codified into the client_identity of the requested TA session.**

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 a TA, then the client_identity must be equal to the TA_identity of the TA subject that is the client.**

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.**

Application Note:

TEE Internal API defines the codification rules of the CA identity.

FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles

- **TSF;**
- **TA_User.**

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

Application Note:

The TA_User role is the TSF running on behalf of a TA, upon request from the REE (by Client Applications) or from other TAs.

8.3.1.2 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.TA_INSTANCE, S.TA_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.TA_INSTANCE and S.RAM_UNIT:**
  - **Flow of I.RUNTIME_DATA from S.TA_INSTANCE to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(S.TA_INSTANCE) is Write or ReadWrite**
  - **Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.TA_INSTANCE is allowed only if S.RAM_UNIT.rights(S.TA_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(REE) 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(REE) 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 TEE control (S.RESOURCE.state = TEE).**

FDP_IFF.1.3/Runtime The TSF shall enforce the **no 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.TA_INSTANCE_SESSION:**
  - **Flow of I.RUNTIME_DATA that are parameter or return values is allowed between S.TA_INSTANCE_SESSION and S.COMM_AGENT**
  - **Flow of I.RUNTIME_DATA that are parameter or return values is allowed between S.TA_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 TEE 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 TEE firmware and software, in integrity and confidentiality for TEE runtime data).

- RAM units can span over several volatile memories, for example, on-chip RAM, off-chip RAM, registers.
- TEE-dedicated RAM units may hold copies of the content of temporary memory references passed by the REE.

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 TEE 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: **TEE and TA runtime objects.**

Application Note:

This operation applies in particular upon:

- Failure detection (cf. FPT_FLS.1)
- TA instance and TA 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 TEE may reside in physically separated parts.

8.3.1.3 Cryptography

FCS_CKM.1/Store Cryptographic key generation

FCS_CKM.1.1/Store	The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **HW_KBKDF** and specified cryptographic key sizes **256 bits** that meet the following: **[NIST 800-108].**

Application Note

This key generation method is used to generate the keys used for Trusted Storage as defined in FCS_COP.1/Store.

FCS_CKM.4/Store Cryptographic key destruction

FCS_CKM.4.1/Store	The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **overwriting with zeros** that meets the following: **none.**

Application Note:

This key destruction method is used to destroy the keys used for Trusted Storage as defined in FCS_COP.1/Store.

FCS_CKM.1/Install Cryptographic key generation

FCS_CKM.1.1/Install	The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **HW_KBKDF** and specified cryptographic key sizes **256 bits** that meet the following: **[NIST 800-108].**

Application Note

This key generation method is used to generate the keys used for Installation as defined in FCS_COP.1/Install.

FCS_CKM.4/Install Cryptographic key destruction

FCS_CKM.4.1/Install	The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **overwriting with zeros** that meets the following: **none.**

Application Note:

This key destruction method is used to destroy the keys used for Installation as defined in FCS_COP.1/ Install.

FCS_COP.1/Auth Cryptographic operation

FCS_COP.1.1/Auth	The TSF shall perform **signature verification** in accordance with a specified cryptographic algorithm **ECDSA** and cryptographic key sizes **256 bits** that meet the following: **[ECDSA].**

Application Note:

This cryptographic operation is used for verifying the authenticity of the following elements:

- TEE firmware and software: when use for TEE firmware and software code protection, ECDSA operation is based on the root-of-trust key for Secure Boot functionality.
- TA code: when use for TA code protection, ECDSA operation is based on an OEM certificate used to sign all TA specific keys.
- CSMC
- TMF
- TA TUI resource

FCS_COP.1/Store Cryptographic operation

FCS_COP.1.1/Store	The TSF shall perform **encryption for confidentiality and integrity** in accordance with a specified cryptographic algorithm **AES-GCM** and cryptographic key sizes **256 bits** that meet the following: **[AES].**

Application Note:

This cryptographic operation is used for protecting the consistency and confidentiality of Trusted Storage data. This operation is based on the root-of-trust key for TEE Trusted Storage functionality.

FCS_COP.1/Install Cryptographic operation

FCS_COP.1.1/Install	The TSF shall perform **encryption for confidentiality and integrity** in accordance with a specified cryptographic algorithm **AES-CTR** and cryptographic key sizes **256 bits** that meet the following: **[AES].**

Application Note:

This cryptographic operation is used for protecting confidentiality, Integrity, Authenticity and consistency of TA during downloading.

FCS_COP.1/Digest Cryptographic operation

FCS_COP.1.1/Digest	The TSF shall perform **hashing operations** in accordance with cryptographic algorithm **SHA-1, SHA-224, SHA-256, SHA-384, SHA-512** and cryptographic key sizes **none** that meet the following: **GP API 1.0, GP API 1.1, GP API 1.2 and [Hash].**

FCS_COP.1/Cipher Cryptographic operation

FCS_COP.1.1/Cipher	The TSF shall perform **symmetric cipher operations** in accordance with cryptographic algorithms **AES-ECB, AES-CBC, AES-CTR, AES-XTS, 3DES-ECB, 3DES-CBC** and cryptographic key sizes **for AES 128, 192, 256 bits, for 3DES 112, 168 bits** that meet the following: **GP API 1.0, GP API 1.1, GP API 1.2 and [AES] or [3DES].**

Application Note:

Supported 3DES key size starts with 112 bits. This size is limited for usage until end of 2025. From 2026, only key sizes of 128 and above are authorized.

FCS_COP.1/MAC Cryptographic operation

FCS_COP.1.1/MAC	The TSF shall perform ***MAC operations*** in accordance with a specified cryptographic algorithm **HMAC, AES-MAC, AES-CMAC, 3DES-MAC** and cryptographic key sizes **for AES 128, 192, 256 bits, for 3DES 112, 128 bits** that meet the following: **GP API 1.0, GP API 1.1, GP API 1.2 and [HMAC], or [CMAC], [AES] or [3DES].**

Application Note:

Supported 3DES key size starts with 112 bits. This size is limited for usage until end of 2025. From 2026, only

key sizes of 128 and above are authorized.

FCS_COP.1/AE Cryptographic operation

FCS_COP.1.1/AE The TSF shall perform **Authenticated Encryption operations (encryption and decryption)** in accordance with a specified cryptographic algorithm **AES-CCM, AES-GCM** and cryptographic key sizes **128, 192, 256 bits** that meet the following: **GP API 1.0, GP API 1.1, GP API 1.2 and [AE].**

FCS_COP.1/Enc Cryptographic operation

FCS_COP.1.1/Enc The TSF shall perform **asymmetric encryption and decryption operations** in accordance with cryptographic algorithm **RSA PKCS1-v1.5 or RSA OAEP** and cryptographic key sizes **from 2048 up to 4096 bits, as multiply of 128 bits** that meet the following: **GP API 1.0, GP API 1.1, GP API 1.2 and [RSA].**

Application Note:

Supported RSA key size starts with 256 bits. However, lengths lower than 2048 are forbidden to use. Sizes between 2048 and 3072 are limited for usage until end of 2030. From 2031, only key sizes of 3072 and above are authorized.

FCS_COP.1/Sig Cryptographic operation

FCS_COP.1.1/Sig The TSF shall **perform signature verification and generation operations** in accordance with cryptographic algorithm **RSA PKCS1-v1.5, RSA PSS, DSA or ECDSA** and cryptographic key sizes **for RSA from 2048 up to 4096 bits, as multiply of 128 bits or for ECDSA 256, 384, 521 bits or for ED25519 256bits** that meet the following: **GP API 1.0, GP API 1.1, GP API 1.2 and [RSA], [DSA], [ECDSA] or [ED25519].**

Application Note:

Supported RSA key size starts with 256 bits. However, lengths lower than 2048 are forbidden to use. Sizes between 2048 and 3072 are limited for usage until end of 2030. From 2031, only key sizes of 3072 and above are authorized.

FCS_COP.1/KEA Cryptographic operation

FCS_COP.1.1/KEA The TSF shall perform **key derivation operations** in accordance with a specified cryptographic algorithm **Diffie-Hellman (DH) and ECDH** and cryptographic key sizes for **DH 2048 bits, as multiply of 8 bits or ECDH 256, 384, 521 bits or for X25519 256bits** that meet the following: **GP API 1.0, GP API 1.1, GP API 1.2 and [DH], [ECDH] or [X25519].**

Application Note:

Supported RSA key size starts with 256 bits. However, lengths lower than 2048 are forbidden to use. Sizes between 2048 and 3072 are limited for usage until end of 2030. From 2031, the usage of this feature is not authorized anymore.

FDP_ACC.1/TA_keys Subset access control

FDP_ACC.1.1/TA_keys	The TSF shall enforce the **TA Keys Access Control SFP** on ○ **Subjects: S.API, S.TA_INSTANCE and any other subject in the TEE** ○ **Objects: OB.TA_KEY** ○ **Operations: OP.USE_KEY, OP.EXTRACT_KEY.**

FDP_ACF.1/TA_keys Security attribute based access control

FDP_ACF.1.1/TA_keys	The TSF shall enforce the **TA Keys Access Control SFP** to objects based on the following: **OB.TA_KEY.usage, OB.TA_KEY.owner, OB_TA_KEY.isExtractable, and S.API.caller.**
FDP_ACF.1.2/TA_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 TA instance that requested the operation to the API owns the key (S.API.caller = OB.TA_KEY.owner)** ▪ **The intended usage of the key (OB.TA_KEY.usage) matches the requested operation** ○ **OP.EXTRACT_KEY is allowed if the following conditions hold:** ▪ **The TA instance that requested the operation to the API owns the key (S.API.caller = OB.TA_KEY.owner)** ▪ **The operation attempts to extract the public part of OB.TA_KEY or the key is extractable (OB.TA_KEY.isExtractable = True).**
FDP_ACF.1.3/TA_keys	The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **none.**
FDP_ACF.1.4/TA_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.TA_INSTANCE or any other subject of the TEE that is not S.API** ○ **Any access to a user key attempted from S.API without valid caller (S.API.caller is undefined)**

Application Note:

This requirement states access conditions to keys through the TEE Internal API only: OP.USE_KEY and OP.EXTRACT_KEY stand for operations of the API.

FDP_ACF.1.3/TA_keys: Note that ownership in the current TEE internal API specification is limited to each TA having access to all, and only to, its own objects.

FMT_MSA.1/TA_keys Management of security attributes

FMT_MSA.1.1/TA_keys	The TSF shall enforce the **TA Keys Access Control SFP** to restrict the ability to **change_default, query, and modify** the security attributes **OB.TA_KEY.usage, OB.TA_KEYS.isExtractable and OB.TA_KEY.owner** to **the following roles:** ○ **change_default, query and modify OB.TA_KEY.usage to TA_User role** ○ **query OB.TA_KEY.owner to the TSF role.**

FMT_MSA.3/TA_keys Static attribute initialization

FMT_MSA.3.1/TA_keys	The TSF shall enforce the **TA Keys Access Control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.
FMT_MSA.3.2/TA_keys	The TSF shall allow the **TA_User role** to specify alternative initial values to override the default values when an object or information is created.

8.3.1.4 Initialization, Operation and Firmware Integrity

FAU_ARP.1 Security alarms

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

Refinement:

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

- Detection of consistency violation of TA data, TA code or TEE data: **at starting of TA, stop starting**.
- Detection of integrity violation of TA code and TEE code: **at starting of TA, stop starting.**
- Detection of TEE firmware integrity violation: **stop booting and reset after timeout**.
- Detection of memory access violation: **raise an alarm**.

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 listed in refinement]** on all objects, based on the following attributes: **[assignment: user data attributes listed in refinement]**.

Refinement:

The TSF shall monitor TEE runtime data, TEE persistent data, TA data and keys and TA code stored in containers controlled by the TSF for **authenticity and consistency errors** on all objects, based on the following attributes: **attributes of TEE runtime data, TEE persistent data, TA data and keys, and TA code.**

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

Refinement:

- Upon detection of authenticity or consistency errors in TEE runtime data or TEE persistent data, the TSF shall **return error code.**
- Upon detection of TA code authenticity or consistency errors, the TSF shall **abort the execution of the TA instance.**
- Upon detection of TA data or TA keys authenticity or consistency errors, the TSF shall:
  - **Not give back any compromised data;**
  - **Return error code.**

Application Note:

This SFR applies to TEE runtime data in volatile memory (this data is not stored in non-volatile memory) and to TEE persistent data, TA data and keys and TA code in both volatile and non-volatile memory.

This SFR is used for both TSF and user data as similar mechanisms are involved to protect the consistency of this data.

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 (as defined in [TEE CORE 1.0], Section 2.2.3) ;**
- **TA code, TA data or TA keys authenticity or consistency failure ;**
- **authenticity or consistency failure of TEE data (in particular TA properties, TEE keys and all security attributes);**
- **TEE firmware, TEE software and some TA code integrity failure ;**
- **TEE initialization failure ;**
- **Unexpected commands.**

Application Note:

The secure state is preserved when each of the aforementioned failure occurs. The actions upon failure are described as follows:

- Device binding failure: device binding failure is detected as “authentication failure” while loading the stored data. Upon such failure, TEE stored data is reset to initial state discarding un-authenticated data.
- Cryptographic operation failure: cryptographic operations of the cryptographic library return results of execution that are checked by internal implementation of GP API. The critical failures are detected and TA execution terminated with *TEE_Panic* procedure.
- Invalid CA requests, in particular bad-formed requests: the requests from invalid CA can be identified to some extent using CA Identity. The identity of the CA to be used is supplied by the REE. The secure state is restored by discarding such invalid requests identified by the checks mentioned above and client is notified with an error condition.
- TA code, TA data or TA keys authenticity or consistency failure: TA code integrity and authenticity is protected by signature verification process. TA persistent data is stored in trusted storage and any other

data that is part of the TA image is protected by signature verification. Same for being detected by cryptographic operation failure. Secure state is preserved by discarding inputs to cryptographic operation and randomizing the memory where such inputs were present.

- Authenticity and consistency of TEE data (in particular TA properties, TEE keys and all security attributes): the secure state of TEE data is restored by discarding the content whose integrity or authenticity is compromised. TA Properties are part of the TA image. Any tampering to the properties of a TA on file system is detected by signature verification. Secure state is preserved by aborting the execution of such TA whose properties do not meet the integrity/authenticity.
- TEE firmware and some TA code integrity failure: TEE Firmware integrity failure is detected during the signature verification on the firmware image. If the integrity check fails, the bootloader will not load TEE firmware and will prevent the device from booting. Some TA code integrity checks lead to a stop of booting in case of failure.
- TEE initialization failure: the failure causes a time-out reset.
- Unexpected commands: TEE does not define any states and handles all valid commands by some checks ensuring that the command can be processed. Unexpected commands (independent of state) are always not served.

FPT_INI.1 TSF initialization

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

Application Note:

This covers for instance code/data that are stored and executed from non-modifiable memory at boot time, the immutable root-of-trust, and other OTP values such as versions and identifiers.

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

	Properties	Elements
1	authenticity and integrity	TEE firmware
2	prevention of downgrade to previous versions	TEE firmware
3	integrity	TEE Storage Root of Trust
4	integrity	TEE identification data

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_INI.1.3 The TOE initialization function shall detect and respond to errors and failures during initialization such that the TOE **is halted.**

FPT_INI.1-4 The TOE initialization function shall only interact with the TSF **in the way of maintaining properties described above** during initialization.

FMT_SMF.1 Specification of Management Functions

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

- **Management of TA 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 to execution** to check the fulfillment of **authenticity of TA code**.

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

8.3.1.5 TEE Identification

FAU_SAR.1 Audit review

FAU_SAR.1.1 The TSF shall provide **all users** with the capability to read the **TEE 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 refers to the TEE identifier. This unique identifier is stored on the TEE at TEE initialization stage. The TEE identifier is included in the binary code, signed, and stored on TEE with binary.

8.3.1.6 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 TA instances such that time stamps are monotonic during the TA instance lifetime.

Application Note:

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

8.3.1.7 Random Number Generator

FCS_RNG.1 Random numbers generation

FCS_RNG.1.1 The TSF shall provide a **hybrid** random number generator that implements:

- **A True Random Number Generator (TRNG): it is the physical source of randomness. Its output is used as the seed for generating random numbers;**
- **A Deterministic Random Bit Generator (DRBG): it takes the output (the seed) from TRNG and applies mathematical computation based on deterministic algorithms to generate the random numbers.**

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

- **The TRNG is evaluated with a set of statistic tests specified by FIPS 140-2, namely the Monobit, Poker, Runs and Longest Run tests;**
- **The DRBG uses the NIST SP800-90A approved algorithm for generating the random numbers based on the seed.**

Application Note:

The correct entropy is ensured by the use of a hardware RNG source complemented by a DRBG in case of hardware failure.

8.3.1.8 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.TA_STORAGE, OB.SRT ;**
- **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.TA_STORAGE.owner, OB.TA_STORAGE.inExtMem, OB.TA_STORAGE.TEE_identity and OB.SRT.TEE_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.TA_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.TA_STORAGE.owner) ;**
  - **OB.TA_STORAGE is bound to the TEE storage**

root of trust OB.SRT (OB.TA_STORAGE.TEE_identity = OB.SRT.TEE_identity) ;

- **If OB.TA_STORAGE is located in external memory accessible to the REE (OB.TA_STORAGE.inExtMem = True) then the object is authenticated and decrypted before load.**

- **OP.STORE of an object to OB.TA_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.TA_STORAGE.owner) ;**
  - **OB.TA_STORAGE is bound to the TEE storage root of trust OB.SRT (OB.TA_STORAGE.TEE_identity = OB.SRT.TEE_identity) ;**
  - **If OB.TA_STORAGE is located in external memory accessible to the REE (OB.TA_STORAGE.inExtMem = True) then the object is signed and encrypted before storage.**

FDP_ACF.1.3/Trusted Storage	The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.
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 TEE (OB.TA_STORAGE.TEE_identity different from OB.SRT.TEE_identity) ;** ○ **Any access to a trusted storage from a subject different from S.API.**

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 **limit of one previous version.**

Application Note:

This SFR enforces atomicity of any write operation [TEE CORE 1.0].

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.TA_STORAGE.owner, OB.TA_STORAGE.inExtMem, OB.TA_STORAGE.TEE_identity and OB.SRT.TEE_identity** to **TA_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 **TA_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.

8.3.2 TEE Debug PP-module

To simplify reading, the color code is not used here.

The additional User introduced by this PP-Module is:

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

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

- OP.AUTHENTICATE: Activation of the debug feature by TEE 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 access to debug facilities of the TEE.
- 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**

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 TEE Debug Administrator is authenticated (S.DEBUG.authenticated = True)**

FDP_ACF.1.3/Debug The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none.**

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

- **Any access to OP.DEBUG without correct authentication of TEE Debug Administrator (S.DEBUG.authenticated = False)**
- **Any access to OP.DEBUG in case of OP.DEBUG session failure.**

FCS_COP.1/Debug Cryptographic operation

FCS_COP.1.1/Debug The TSF shall perform **authentication of the TEE Debug Administrator or the actor acting on his behalf** in accordance with a specified cryptographic algorithm **ECDSA** and cryptographic key sizes **512** that meet the following: **[ECDSA].**

FMT_SMR.1/Debug Security roles

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

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

Application Note:

The TEE 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 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 TEE 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)**

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 TEE power-off** ○ **after OP.DEBUG session closing.**

8.3.3 TEE Time and Rollback PP-module

To simplify reading, the color code is not used here.

8.3.3.1 Rollback Protection

FDP_SDI.2/Rollback Stored data integrity monitoring and action

FDP_SDI.2.1/Rollback	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 TEE rollback detection data, TEE runtime data, TEE persistent data, TA data and keys, and TA code stored in containers controlled by the TSF for **integrity errors** on all objects, based on the following attributes: **attributes of TEE rollback detection data, TEE runtime data integrity, TEE persistent data integrity, TA data and keys integrity, and TA code authenticity.**
FDP_SDI.2.2/Rollback	Upon detection of a data integrity error, the TSF shall **[assignment: action to be taken].** *Refinement:* ○ Upon detection of integrity errors in TEE rollback detection data, TEE runtime data or TEE persistent data, the TSF shall **behave in a manner that does not depend on the compromised data** ○ Upon detection of TA code integrity errors, the TSF shall **abort the execution of the TA instance** ○ Upon detection of TA data or TA keys integrity errors, the TSF shall: ▪ **Not provide any compromised data,** ▪ **Behave in a manner that does not depend on the compromised data**

Application Note:

This requirement adds integrity monitoring to the FDP_SDI.2 defined in the core TEE PP. Rollback detection is ensured by rollback detection data and by integrity failure detection.

FPT_FLS.1/Rollback Failure with preservation of secure state

FPT_FLS.1.1/Rollback	The TSF shall preserve a secure state when the following types of failures occur: ○ **TA code and data integrity failure** ○ **TEE persistent data integrity failure.**

Application Note:

This requirement is a complement to FPT_FLS.1.

8.3.3.2 TA Persistent Time

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 TA instances such that: ○ Time stamps are persistent over TEE reset ○ Time stamps are monotonic between two 'time setting' operations performed by any instance of the TA. The TSF shall invalidate any persistent time that does not meet the monotonicity property.

Application Note:

The refinement provides the meaning of the expected reliability.

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 **TA persistent time** to **any instance of the TA**.

Application Note:

The 'time setting' operation will only affect the persistent time value of the TA 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 function: **'time setting' operation for TA persistent time.**

Application Note:

The 'time setting' operation will only affect the persistent time value of the TA performing the operation.

8.3.4 Trusted User Interface SFRs

To simplify reading, the color code is not used here.

Security Policy - TEE Trusted User Interface Access Control SFP

This PP-Module requires a security access control policy to peripherals and TUI data, called **TEE Trusted User Interface Access Control SFP**, to enforce the correct behavior of the TUI functionality and the interactions between the peripherals and the TAs.

The subjects, objects, security attributes and operations of this policy are the following:

Subjects: The active entities are the TUI system itself, the TAs and the peripherals.

Objects: persistent state and a transient state of the system.

The minimum persistent state of the system consists of:

- peripheral_list: a list of unique identifiers of authorized input/output peripherals

The minimum transient state of the system consists of:

- calling_TA: the current instance of TA that requests an interaction with peripherals
- out_data: data sent from the calling TA to a locked peripheral
- in_data: data received by the calling TA from a locked peripheral

Security attributes:

- peripheral.status: the status “locked”, “unlocked” or “exclusively-locked” of the peripheral PR
- peripheral.type: the type “IN”, “OUT” or “I/O” of the peripheral PR
- peripheral.ownership: the owner “TEE” or “REE” of the peripheral PR
- peripheral.class : the class of the peripheral: TEE-only, Shareable, REE-only
- peripheral.exclusive_access: the flag “Yes” or “No” of the peripheral PR identifying whether PR supports exclusive access
- peripheral.ownership_changed: a flag “Yes” or “No” showing if the peripheral’s ownership has been temporarily transferred (from the REE to the TEE)

Operations:

- peripheral_discovery: return the list p_list of peripherals available for the calling TA; p_list should be a subset of peripheral_list
- lock_peripherals(p_list): lock all peripherals in p_list for exclusive access; it ensures that for all peripherals p in p_list, p.status == exclusively-locked and p.ownership == TEE
- unlock_peripherals(p_list): unlock or release the peripherals in p_list; it ensures that for all peripherals p in p_list, p.status == unlocked and p.ownership is set to the peripheral’s initial owner
- send_data(o, p): send output data o from the calling TA to the locked peripheral p
- receive_data(i, p): send input data i from the locked peripheral p to the calling TA

Internal operations:

- capture_input(p, d): capture input data d using the peripheral p
- present_output(p, d): present/display output data d using the peripheral p
- transfer_ownership(p): transfer the ownership of peripheral p, i.e. from “REE” to “TEE”;
- If p.ownership == REE before running transfer_ownership(p), then p.ownership == TEE and p.ownership_changed == Yes after running it;

Application Note:

- The operation transfer_ownership() is by default on each peripheral device on TEEgris implementation.
- These operations must be executed in a specific order to provide the expected service.
  - The symbol “;” is used to indicate a sequence of operations.
  - The symbol “+” attached to an operation is used to indicate that the operation can be run one or more times.
  - The symbol “*” attached to an operation is used to indicate that the operation is optional, i.e. it can be run zero or more times.

FDP_ACC.1/TUI Subset access control

FDP_ACC.1.1/TUI The TSF shall enforce the **TEE Trusted User Interface Access Control SFP** on:

- **Subjects: TUI system, TAs, peripherals**
- **Objects:**
  - **Persistent objects: peripheral_list;**
  - **Transient objects: calling_TA, in_data, out_data;**
- **Operations:**
  - **lock_peripherals, unlock_peripherals, peripheral_discovery, send_data, receive_data;**
  - **internal operations: transfer_ownership, capture_input, present_output,;**

FDP_ACF.1/TUI Security attribute based access control

FDP_ACF.1.1/TUI The TSF shall enforce the **TEE Trusted User Interface Security attribute based Access Control SFP** to objects based on the following:

- **peripheral.status (“locked”, “unlocked” or “exclusively-locked”)**
- **peripheral.ownership ("TEE” or “REE”)**
- **peripheral.type ("IN”, “OUT” or “I/O”)**
- **peripheral.class ("TEE-only”, “Shareable”, “REE-only”)**
- **peripheral.exclusive_access ("Yes” or “No”)**

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

- **rule-lock-peripherals:**

  lock_peripherals (pls) is allowed

  if pls is a subset of peripheral_list and for all peripheral in pls peripheral.status == unlocked and peripheral.ownership == TEE and peripheral.exclusive_access == YES
- **rule-transfer-owner-and-lock-peripherals:**

  transfer_ownership()+; lock_peripherals (pls) is allowed

  if pls is a subset of peripheral_list and for all peripheral in pls peripheral.status == unlocked and peripheral.exclusive_access == YES and either peripheral.ownership == TEE or (peripheral.ownership == REE and peripheral.class == Shareable)
- **rule-unlock-peripherals:**

  unlock_peripherals(pls) is allowed

  if pls is a subset of peripheral_list and for all peripherals in pls either peripheral.status == locked or exclusively-locked, and peripheral.ownership_changed == No
- **rule-transfer-owner-and-unlock-peripherals:**

  transfer_ownership(peripheral)+; unlock_peripherals (pls) is allowed

if *pls* is a subset of peripheral_list and for all peripheral in *pls* either peripheral.status == locked or exclusively-locked, and peripheral.ownership_changed == Yes

- **rule-transfer-owner-and-unlock-peripherals:**

  transfer_ownership(peripheral)+; switch_SI_state(si)+; unlock_peripherals (pls) is allowed

  if *pls* is a subset of peripheral_list and for all peripheral in *pls* either peripheral.status == locked or exclusively-locked, and peripheral.ownership_changed == Yes and si_map(peripheral).state == ON

- **rule-send-data:**

  send_data(out_data, peripheral); present_data(od, peripheral); is allowed if peripheral.status == locked and peripheral.type == OUT or I/O

Application Note:

The output data out_data sent by a TA to be presented to users on an output peripheral may be transformed by the peripheral before presenting/displaying it.

rule-receive-data:

capture_data(peripheral, in_data); receive_data (in_data, peripheral); is allowed if peripheral.status == locked and peripheral.type == In or I/O

FDP_ACF.1.3/TUI The TSF shall explicitly authorise access of subjects to objects based on the following additional rules:

- **None.**

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

- **rule-deny-lock-peripherals**: if any of the conditions stipulated for lock_peripherals does not hold
- **rule-deny-unlock-peripherals**: if any of the conditions stipulated for unlock_peripherals does not hold
- **rule-deny-send-data**: if any of the conditions stipulated for send_data does not hold
- **rule-deny-receive-data**: if any of the conditions stipulated for receive_data does not hold

FDP_RIP.1/TUI Subset residual information protection

FDP_RIP.1.1/TUI 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:

- **out_data, in_data, as per FDP_ACC.1/TUI upon unlock_peripherals**
- **trusted path as per FDP_TRP.1/TUI upon unlock_peripherals]**

FMT_MSA.1/TUI Management of security attributes

FMT_MSA.1.1/TUI The TSF shall enforce the **TEE Trusted User Interface Access Control SFP** to restrict the ability to **query and modify** the security attributes:

- **peripheral.status**
- **peripheral.ownership**
- **peripheral.type**
- **peripheral.class**
- **peripheral.exclusive_access**
- **peripheral.ownership_changed**

to **the TUI system.**

FMT_MSA.2/TUI Secure security attributes

FMT_MSA.2.1/TUI The TSF shall ensure that only secure values are accepted for **the following security attributes:**

- **peripheral.status == locked or unlocked or exclusively-locked**
- **peripheral.ownership == TEE or REE**
- **peripheral.exclusive_access == Yes or No**
- **peripheral.type == IN or OUT or I/O**
- **peripheral.class == TEE-only or Shareable or REE-only**
- **peripheral.ownership_changed == Yes or No**

FMT_MSA.3/TUI Static attributes initialisation

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

FMT_MSA.3.2/TUI The TSF shall allow **no role** to specify alternative initial values to override the default values when an object or information is created.

FPT_FLS.1/TUI Failure with preservation of secure state

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

- **a TUI operation is halted**
- **a panic occurs**
- **an operation is denied as per FDP_ACF.1/TUI**
- **a trusted path tampering has been detected as per FTP_TPR.1/TUI**
- **cryptographic operation failure on TUI resource.**

Application Note:

As defined in [CC1], secure state stands for “state in which the TSF data are consistent and the TSF continues

correct enforcement of the SFRs".

FTP_TRP.1/TUI

FTP_TRP.1.1/TUI	The TSF shall provide a communication path between itself and **local** users that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from **modification and disclosure by the REE and/or TAs.** ***Application Note:*** In practice, the trusted path provides protected communication from and to the peripherals. "Local users" stands for a user interface.
FTP_TRP.1.2/TUI	The TSF shall permit **the TSF** to initiate communication via the trusted path.
FTP_TRP.1.3/TUI	The TSF shall require the use of the trusted path for **exchanging, i.e. sending and receiving, data to and from controlled peripherals.**

8.4 Security Assurance Requirements

This Security Target conforms to Protection Profile and provides a set of Security Assurance Requirements (SARs) that consists of the EAL 2 predefined package augmented with the extended component AVA_VAN_AP.3, which requires Enhanced-basic attack potential (cf. definition in PP TEE).

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

AVA_VAN_AP.3 TEE 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, security architecture description **and the following parts of the TSF implementation representation: source code (except cryptographic implementation) and cryptographic documentation** 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.

8.5 Security Requirements Rationale

8.5.1 Rationale Objectives/SFRs

	O.CA_TA_IDENTIFICATION	O.KEYS_USAGE	O.TEE_ID	O.INITIALISATION	O.INSTANCE_TIME	O.OPERATION	O.RNG	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TEE_ISOLATION	O.TRUSTED_STORAGE	O.DEBUG	O.ROLLBACK_PROTECTION	O.TA_PERSISTENT_TIME	O.PERIPHERAL_INITIALISATION	O.PROTECTED_COMMUNICATION_CHANNEL	O.PREVENT_RESIDUAL_DATA	O.DATA_ACCESS	O.FUNCTION_ACCESS	O.SAFE_RELEASE
FIA_ATD.1	x					x																	
FIA_UID.2	x					x																	
FIA_USB.1	x					x																	
FMT_SMR.1		x				x																	
FDP_IFC.2/Runtime						x		x	x		x		x										
FDP_IFF.1/Runtime						x		x	x		x		x										

	O.CA_TA_IDENTIFICATION	O.KEYS_USAGE	O.TEE_ID	O.INITIALISATION	O.INSTANCE_TIME	O.OPERATION	O.RNG	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TEE_ISOLATION	O.TRUSTED_STORAGE	O.DEBUG	O.ROLLBACK_PROTECTION	O.TA_PERSISTENT_TIME	O.PERIPHERAL_INITIALISATION	O.PROTECTED_COMMUNICATION_CHANNEL	O.PREVENT_RESIDUAL_DATA	O.DATA_ACCESS	O.FUNCTION_ACCESS	O.SAFE_RELEASE	
FDP_ITT.1/Runtime								x	x														
FDP_RIP.1/Runtime								x															
FPT_ITT.1/Runtime								x	x			x											
FCS_CKM.1/Store		x									x	x		x									
FCS_CKM.4/Store		x									x	x		x									
FCS_CKM.1/Install		x								x													
FCS_CKM.4/Install		x								x													
FCS_COP.1/Auth				x						x													
FCS_COP.1/Store		x									x	x		x									
FCS_COP.1/Install		x								x													
FCS_COP.1/Digest		x																					
FCS_COP.1/Cipher		x																					
FCS_COP.1/MAC		x																					
FCS_COP.1/AE		x																					
FCS_COP.1/Enc		x																					
FCS_COP.1/Sig		x																					
FCS_COP.1/KEA		x																					
FDP_ACC.1/TA_keys		x				x																	

	O.CA_TA_IDENTIFICATION	O.KEYS_USAGE	O.TEE_ID	O.INITIALISATION	O.INSTANCE_TIME	O.OPERATION	O.RNG	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TEE_ISOLATION	O.TRUSTED_STORAGE	O.DEBUG	O.ROLLBACK_PROTECTION	O.TA_PERSISTENT_TIME	O.PERIPHERAL_INITIALISATION	O.PROTECTED_COMMUNICATION_CHANNEL	O.PREVENT_RESIDUAL_DATA	O.DATA_ACCESS	O.FUNCTION_ACCESS	O.SAFE_RELEASE
FDP_ACF.1/TA_keys		x				x																	
FMT_MSA.1/TA_keys		x				x																	
FMT_MSA.3/TA_keys		x				x																	
FAU_ARP.1						x																	
FDP_SDI.2						x			x	x		x		x									
FPT_FLS.1				x							x			x									
FPT_INI.1			x	x										x				x					
FMT_SMF.1		x				x					x			x									
FPT_TEE.1										x													
FAU_SAR.1			x																				
FAU_STG.1			x																				
FPT_STM.1/Instance time					x																		
FCS_RNG.1			x				x																
FDP_ACC.1/Trusted Storage						x					x			x									
FDP_ACF.1/Trusted Storage						x					x			x									
FDP_ROL.1/Trusted Storage														x									
FMT_MSA.1/Trusted Storage						x					x			x									
FMT_MSA.3/Trusted Storage						x					x			x									

	O.CA_TA_IDENTIFICATION	O.KEYS_USAGE	O.TEE_ID	O.INITIALISATION	O.INSTANCE_TIME	O.OPERATION	O.RNG	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TEE_ISOLATION	O.TRUSTED_STORAGE	O.DEBUG	O.ROLLBACK_PROTECTION	O.TA_PERSISTENT_TIME	O.PERIPHERAL_INITIALISATION	O.PROTECTED_COMMUNICATION_CHANNEL	O.PREVENT_RESIDUAL_DATA	O.DATA_ACCESS	O.FUNCTION_ACCESS	O.SAFE_RELEASE
FDP_ITT.1/Trusted Storage														x									
FDP_ACC.1/Debug						x									x								
FDP_ACF.1/Debug						x									x								
FCS_COP.1/Debug		x													x								
FMT_SMR.1/Debug						x									x								
FIA_UID.2/Debug						x									x								
FIA_ATD.1/Debug						x									x								
FIA_USB.1/Debug						x									x								
FIA_UAU.2/Debug						x									x								
FIA_UAU.6/Debug						x									x								
FDP_SDI.2/Rollback						x										x							
FPT_FLS.1/Rollback						x										x							
FPT_STM.1/Persistent Time																	x						
FMT_MTD.1/Persistent Time																	x						
FMT_SMF.1/Persistent Time																	x						
FDP_ACC.1/TUI																			x		x	x	
FDP_ACF.1/TUI																			x		x	x	x
FDP_RIP.1/TUI																				x			

	O.CA_TA_IDENTIFICATION	O.KEYS_USAGE	O.TEE_ID	O.INITIALISATION	O.INSTANCE_TIME	O.OPERATION	O.RNG	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TEE_ISOLATION	O.TRUSTED_STORAGE	O.DEBUG	O.ROLLBACK_PROTECTION	O.TA_PERSISTENT_TIME	O.PERIPHERAL_INITIALISATION	O.PROTECTED_COMMUNICATION_CHANNEL	O.PREVENT_RESIDUAL_DATA	O.DATA_ACCESS	O.FUNCTION_ACCESS	O.SAFE_RELEASE
FMT_MSA.1/TUI																			x			x	
FMT_MSA.2/TUI																			x			x	
FMT_MSA.3/TUI																			x			x	
FPT_FLS.1/TUI																					x		x
FTP_TRP.1/TUI																			x		x	x	

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

- FIA_ATD.1 enforces the management of the Client and TA 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 TA before any action, thus allowing access to services and data to authorized users only.
- FIA_USB.1 enforces the association of the user identity with 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 fulfil the objective:

- All FCS_COP.1 instances allow to specify the cryptographic operations in the scope of the evaluation.
- FCS_CKM.1/Store, FCS_CKM.4/Store, FCS_CKM.1/Install and FCS_CKM.4/Install specify how to generate and destroy the keys used for Trusted Storage and Installation protection, enforcing their controlled usage.
- FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys, FMT_SMR.1 and FMT_SMF.1 state the key access policy, which grants access to the owner of the key only.

Application Note

The SFRs FCS_CKM.1/Store and FCS_CKM.4/Store contribute to the enforcement of O.KEYS_USAGE and so have been added in this rationale.

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

- FAU_SAR.1 enforces TEE identifier access capabilities.

- FAU_STG.1 enforces TEE identifier storage capabilities.
- FPT_INI.1 enforces the integrity of TEE identification, and it states the behaviour in case of failure.
- FCS_RNG.1 enforces statistical uniqueness of the TEE identification data if it is generated on the TOE.

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

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

O.INSTANCE_TIME The following requirement fulfils the objective:

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

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

- FAU_ARP.1 states the TEE responses to potential security violations.
- FDP_SDI.2 enforces the monitoring of consistency and authenticity of TEE data and TA, and it states the behaviour in case of failure.
- FIA_ATD.1, FIA_UID.2 and FIA_USB.1 ensure that actions are performed by identified users.
- FMT_SMR.1 states the two operational roles enforced by the TEE.
- 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 TA storage.
- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state the policy for controlling access to TA and TEE execution spaces.
- FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys and FMT_SMF.1 state the key access policy.

Rationale specific to the Time and Rollback PP-Module:

- FDP_SDI.2/Rollback enforces the monitoring of integrity of TEE data and TA, and it states the behaviour in case of failure (it completes FDP_SDI.2).
- FPT_FLS.1/Rollback states the complementary abnormal situations that have to lead to a secure state (it completes FPT_FLS.1).

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 TEE if this feature is not disabled.

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

O.RUNTIME_CONFIDENTIALITY The following requirements contribute to fulfil 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 TEE and TA data that is transferred between resources.
- FDP_RIP.1/Runtime states resource clean up policy.

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

- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state TEE and TA 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 TEE and TA data that is transferred between resources.
- FDP_SDI.2 monitors the authenticity and consistency of TEE code, the TEE runtime data, the TA code, and the TA data and keys, and states the response upon failure.

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

- FDP_SDI.2 enforces the consistency and authenticity of TA code during storage.
- FPT_TEE.1 enforces the check of authenticity of TA code prior to execution.
- FCS_COP.1/Auth states the cryptography used to verify the authenticity of TA code. FCS_COP.1/Install states that TA is also protected in confidentiality during its loading. FCS_CKM.1/Install and FCS_CKM.4/Install specify how the installation keys are generated and destroyed. To have derived keys enforces the security of this algorithm and the protection of TA code.

Application Note

The SFRs FCS_CKM.1/Install and FCS_CKM.4/Install contribute to the enforcement of O.TA_AUTHENTICITY in complement to FCS_COP.1/Install and so have been added in this rationale.

O.TA_ISOLATION The following requirements contribute to fulfil 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 TA storage.
- FCS_COP.1/Store state the cryptographic algorithm used for Trusted Storage to ensure confidentiality and authenticity of TA data. FCS_CKM.1/Store and FCS_CKM.4/Store specify how the keys are generated and destroyed. To have derived keys enforces the security of this algorithm and the protection of TA data.
- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state the policy for controlling access to TA execution space.
- FPT_FLS.1 enforces TA isolation by maintaining a secure state, particularly in case of panic states.

Application Note

The SFRs FCS_CKM.1/Store and FCS_CKM.4/Store contribute to the enforcement of O.TA_ISOLATION in complement to FCS_COP.1/Store and so have been added in this rationale.

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

- FCS_COP.1/Store states the cryptography used to protect consistency and confidentiality of the TEE data in external memory. FCS_CKM.1/Store and FCS_CKM.4/Store specify how the keys are generated and destroyed. To have derived keys enforces the security of this cryptography and the protection of TEE data.
- FDP_SDI.2 monitors the authenticity and consistency of TEE persistent data and states the response upon failure.
- FPT_ITT.1/Runtime enforces secure transmission and storage of TEE persistent data.

Application Note

The SFRs FCS_CKM.1/Store and FCS_CKM.4/Store contribute to the enforcement of O.TEE_DATA_PROTECTION in complement to FCS_COP.1/Store and so have been added in this rationale.

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

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

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

- FCS_COP.1/Store states the cryptography used to protect integrity and confidentiality of the TA data in external memory. FCS_CKM.1/Store and FCS_CKM.4/Store specify how the keys are generated. To have derived keys enforces the security of this cryptography and the protection of TA data.
- 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 TA trusted storage and protecting the confidentiality of data.
- FDP_SDI.2 enforces the consistency and authenticity of the trusted storage.
- FPT_INI.1 enforces the integrity of TEE identification and storage root of trust, and it states the behaviour in case of failure.
- FDP_ITT.1/Trusted Storage ensures protection against disclosure of TEE and TA data that is transferred between resources.
- FPT_FLS.1 maintains a secure state.

Application Note

The SFRs FCS_CKM.1/Store and FCS_CKM.4/Store contribute to the enforcement of O.TRUSTED_STORAGE in complement to FCS_COP.1/Store and so have been added in this rationale.

O.DEBUG The following requirements contribute to fulfil 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 TEE Debug Administrator only.
- FCS_COP.1/Debug allows to specify the cryptographic operations used for authenticating TEE Debug Administrator.

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

- FDP_SDI.2/Rollback states the behaviour of the TEE upon integrity failure (thus rollback).
- FPT_FLS.1/Rollback enforces the detection of integrity failure (thus rollback detection).

O.TA_PERSISTENT_TIME The following requirements fulfil the objective:

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

O.PERIPHERAL_INITIALISATION This objective is fulfilled by the TEE through the following requirement:

- FPT_INI.1 enforces the initialisation of peripherals through a secure process including the verification of the authenticity of the peripheral firmware and the integrity of the peripheral's initialisation code and data.

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

- FDP_ACC.1/TUI and FDP_ACF.1/TUI define the policy that controls the interaction between

the peripherals and the TAs and that governs the access to the communication channel that carries exchanged data.

- FMT_MSA.1/TUI, FMT_MSA.2/TUI, and FMT_MSA.3/TUI state the policy for controlling access to the communication channel.
- FTP_TRP.1/TUI enforces a trusted path from/to the peripherals, effectively ensuring the protection of exchanged data.

O.PREVENT_RESIDUAL_DATA The following requirement contributes to fulfilling the objective:

- FDP_RIP.1/TUI states the resource clean up policy and ensures that residual data is made unavailable at the end of any operation involving a peripheral.

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

- FDP_ACC.1/TUI and FDP_ACF.1/TUI define the policy that controls the interaction between the peripherals and the TAs and that governs the access to the data exchanged with the peripheral.
- FTP_TRP.1/TUI enforces a trusted path from/to the peripherals, effectively ensuring the protection of exchanged data.
- FPT_FLS.1/TUI ensures that the secure access to the data exchanged with a peripheral is preserved even in cases of failure or detection of tampering of the trusted path.

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

- FDP_ACC.1/TUI and FDP_ACF.1/TUI define the policy governing the communication between the TOE and a peripheral.
- FMT_MSA.1/TUI, FMT_MSA.2/TUI, and FMT_MSA.3/TUI state the policy for controlling access to the communication channel and the exchanged data.
- FTP_TRP.1/TUI enforces a trusted path from/to the peripherals, effectively ensuring the protection of exchanged data.

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

- FDP_ACF.1/TUI states the policy for securely unlocking, i.e. releasing peripherals.
- FPT_FLS.1/TUI ensures that peripherals are safely released even in the case of a failure or other external events.

8.5.2 Dependencies

8.5.2.1 SFRs Dependencies

Requirements	CC Dependencies	Satisfied Dependencies	Rationale for exclusion of 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)	-
FDP_IFF.1/Runtime	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/Runtime	The dependency to FMT_MSA.3 is discarded. There is no management

Requirements	CC Dependencies	Satisfied Dependencies	Rationale for exclusion of dependencies
			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.
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	-	-
FCS_CKM.1/Store	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_COP.1/Store FCS_CKM.4/Store	-
FCS_CKM.4/Store	(FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1)	FCS_CKM.1/Store	-
FCS_CKM.1/Install	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_COP.1/Install FCS_CKM.4/Install	-
FCS_CKM.4/Install	(FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1)	FCS_CKM.1/Install	-
FCS_COP.1/Auth	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	-	The dependency to FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2 is discarded. The TEE keys used for TEE firmware integrity and TA code integrity in FCS_COP.1/Auth are set during manufacturing. The dependency to FCS_CKM.4 is discarded. The TEE storage root of trust used for cryptographic operations in FCS_COP.1/Auth is not required to be changed or destroyed during the end-usage phase.
FCS_COP.1/Store	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	FCS_CKM.1/Store FCS_CKM.4/Store	-

Requirements	CC Dependencies	Satisfied Dependencies	Rationale for exclusion of dependencies
	and (FCS_CKM.4)		
FCS_COP.1/Install	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1/Install FCS_CKM.4/Install	-
FCS_COP.1/Digest	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	-	This algorithm doesn't imply keys, so no key has to be generated or imported nor destroyed.
FCS_COP.1/Cipher	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	-	The dependency to FCS_CKM.4 is discarded. Key storage and erasing are managed by TA developer.
FCS_COP.1/MAC	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	-	The dependency to FCS_CKM.4 is discarded. Key storage and erasing are managed by TA developer.
FCS_COP.1/AE	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	-	The dependency to FCS_CKM.4 is discarded. Key storage and erasing are managed by TA developer.
FCS_COP.1/Enc	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	-	The dependency to FCS_CKM.4 is discarded. Key storage and erasing are managed by TA developer.
FCS_COP.1/Sig	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	-	The dependency to FCS_CKM.4 is discarded. Key storage and erasing are managed by TA developer.
FCS_COP.1/KEA	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	-	The dependency to FCS_CKM.4 is discarded. Key storage and erasing are managed by TA developer.
FDP_ACC.1/TA_keys	(FDP_ACF.1)	FDP_ACF.1/TA_keys	-
FDP_ACF.1/TA_keys	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/TA_keys, FMT_MSA.3/TA_keys	-
FMT_MSA.1/TA_keys	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1)	FDP_ACC.1/TA_keys, FMT_SMR.1, FMT_SMF.1	-

Requirements	CC Dependencies	Satisfied Dependencies	Rationale for exclusion of dependencies
	and (FMT_SMR.1)		
FMT_MSA.3/TA_keys	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/TA_keys, FMT_SMR.1	-
FAU_ARP.1	(FAU_SAA.1)	-	The dependency to FAU_SAA.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.
FDP_SDI.2	No dependencies	-	-
FPT_FLS.1	No dependencies	-	-
FPT_INI.1	No dependencies	-	-
FMT_SMF.1	No dependencies	-	-
FPT_TEE.1	No dependencies	-	-
FAU_SAR.1	(FAU_GEN.1)	-	The dependency to FAU_GEN.1 is discarded. This dependency is discarded because the only audit record considered is the TEE identifier and this identifier is set before TOE delivery and is non-modifiable afterwards.
FAU_STG.1	(FAU_GEN.1)	-	The dependency to FAU_GEN.1 is discarded. This dependency is discarded because the only audit record considered is the TEE identifier and this identifier is set before TOE delivery and is non-modifiable afterwards.
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	-

Requirements	CC Dependencies	Satisfied Dependencies	Rationale for exclusion of dependencies
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)	FDP_ACC.1/Trusted Storage, FMT_SMR.1, FMT_SMF.1	-
FMT_MSA.3/Trusted Storage	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/Trusted Storage, FMT_SMR.1,	-
FDP_ITT.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted Storage	-
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	The dependency to FMT_MSA.3 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.
FCS_COP.1/Debug	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	-	The dependency to FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2 is discarded. The TEE Debug authentication key used for authenticating TEE Debug Administrator in FCS_COP.1/Debug is set during manufacturing. It cannot be changed during the end-usage phase. The dependency to FCS_CKM.4 is discarded. The TEE Debug authentication key used for TEE Debug Administrator authentication in

Requirements	CC Dependencies	Satisfied Dependencies	Rationale for exclusion of dependencies
			FCS_COP.1/Debug is not required to be changed or destroyed during the end-usage phase.
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	-	-
FDP_SDI.2/Rollback	No dependencies	-	-
FPT_FLS.1/Rollback	No dependencies	-	-
FPT_STM.1/Persistent Time	No dependencies	-	-
FMT_MTD.1/Persistent Time	(FMT_SMF.1) and (FMT_SMR.1)	FMT_SMF.1, FMT_SMR.1	-
FMT_SMF.1/Persistent Time	No dependencies	-	-
FDP_ACC.1/TUI	(FDP_ACF.1)	FDP_ACF.1/TUI	-
FDP_ACF.1/TUI	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/TUI, FMT_MSA.3/TUI	-
FDP_RIP.1/TUI	No dependencies	-	-
FMT_MSA.1/TUI	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.1/TUI	Dependency to FMT_SMF.1 and FMT_SMR.1 is discarded because the operations can only be performed by the TUI itself and no other role is maintained.
FMT_MSA.2/TUI	(FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	FDP_ACC.1/TUI, FMT_MSA.1/TUI	Dependency to FMT_SMR.1 is discarded because the operations can only be performed by the TUI itself and no other role is maintained.
FMT_MSA.3/TUI	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/TUI	Dependency to FMT_SMR.1 is discarded because the operations can only be performed by

Requirements	CC Dependencies	Satisfied Dependencies	Rationale for exclusion of dependencies
			the TUI itself and no other role is maintained.
FPT_FLS.1/TUI	No dependencies	-	-
FTP_TRP.1/TUI	No dependencies	-	-

8.5.2.2 SARs Dependencies

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

8.5.3 Rationale for Security Assurance Requirements

The assurance level defined in this Security Target corresponds to the Protection Profile requirement.

It consists of the predefined assurance package EAL 2 augmented with AVA_VAN_AP.3 in order to reach the Enhanced-basic attack potential as defined in Annex A.

This EAL 2+ permits a developer to gain sufficient assurance from positive security engineering based on good TEE commercial development practices that are compatible with industry constraints, particularly the life cycle of TEE and TEE-enabled devices. The developer has to provide evidence of security engineering at design, testing, guidance, configuration management and delivery levels as required by EAL 2. In order to cope with the high exposure of the TEE and the interest that TEE-enabled 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 attackers' profiles in the field.

By comparison, the standard component AVA_VAN.2 provides a good level of assurance against SW attacks, for instance mobile application malware that is spreading through uncontrolled application stores, exploiting already known SW vulnerabilities. Standard AVA_VAN.2 is well-fit for devices managed within a controlled environment for services which the end user may not have any interest in attacking.

The definition of a specific attack potential scale to be used for AVA_VAN_AP.3 is motivated by additional assurance with protection against easily spreadable attacks that may result from costly vulnerability identification. Such attack paths have been used in some cases against mobile devices, and are common in market segments such as game consoles or TV boxes, where the expected return on investment is higher, and in which the end user has an interest to perform the exploit. In order to reach this goal, the attack potential calculation method to be used for the TEE splits the attack quotation into two phases, identification and exploitation, and defines the attack potential as the sum of identification and exploitation points. The Enhanced- basic attack potential is comparable to the level defined in the JIL's attack quotation table for secure elements.

The 'omponents 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.

9 TOE Summary Specification

Secure Boot

The TEE implements a Secure Boot chain ensuring the secure initialization of the TEE.

At cold boot, the Boot ROM (BROM) code checks the integrity of the ECDSA root-of-trust key for Secure Boot; it then loads the94ootloaderr module and checks its integrity and authenticity by verifying its signature (ECDSA384) with the ECDSA key.

Then, the Exynos Primary Bootloader (EPBL) verifies the signature of:

- The EL3/ATF, loaded in secure external DRAM;

Then, the EL3 verifies the signature of:

- The Secure OS, loaded in secure external DRAM;
- The S-boot/LK, loaded in normal external DRAM.

After the Secure OS image is verified, the execution is transferred from S-boot/LK to the secure OS. The TEE completes the initialization at this stage. In case an integrity violation is detected boot processing, then booting operations are stopped and the TEE is reset after a timeout.

Fulfils the SFRs: FPT_INI.1, FAU_ARP.1, FPT_FLS.1, FCS_COP.1/Auth, FCS_COP.1/Digest

Volatile memory segmentation and isolation

The TEE allows the segmentation of volatile memory (SRAM and DRAM) in areas to which restricted access rights and status are associated.

In particular, for SRAM memory domains can be defined and configured as “secure” or “non-secure”. This is handled by EPBL.

For the external DRAM, the EPBL/EL3 configures the hardware MPU module to define the domains and divide the memory in regions, each region having its own permission for the corresponding domain.

Fulfils the SFRs: FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FAU_ARP.1.

TEE resources access control

The TEE ensures access control to its internal hardware resources (e.g., TRNG, EMMC module, etc.) based on the TrustZone Aware AXI Bus and a configuration Bus Protection Device Access Permission Control module (APC). This solution allows gathering bus masters into domains and associating each domain to permissions. Bus slave accesses are then controlled through those permissions.

This solution also ensures that during data transmission between two components, no manipulation is possible by a third component that would allow the disclosure or modification of transferred data.

The permission settings reside in Always On (AON) power domain preventing the need for backup/restore while device is suspended/resumed.

From a TA point of view, each TA is associated to a profile defining the TEE resources it will be able to access.

Fulfils the SFRs: FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FAU_ARP.1, FDP_ITT.1.1/Runtime, FPT_ITT.1.1/Runtime

TEE debug access control

The TEE ensures access control to its debug availability based on the cryptographic access control. Any debug attempts will be restricted upon the OTP availability. Once OTP related Secure JTAG is disabled, cryptographic operation as challenge-response should be performed to access debug channel. This cryptographic access control will be given at every attempt to access per device.

Fulfils the SFRs: 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

TEE execution monitoring

The TZ daemon is responsible for checking the Secure World/Normal World components stability. Depending on configuration, if TZ daemon detects critical failure in Secure World, it panics and causes either timeout or device hardware reboot. On Secure World restart all currently running tAs are terminated and cAs receive appropriate notification according TEE Client API specification. Later, the TZ daemon is restarted by the Rich OS and it re-establishes communications with TEE. cAs have to re-connect to appropriate TEE Secure Services. All previously allocated resources and ongoing operations are terminated.

Fulfils the SFRs: FAU_ARP.1, FPT_FLS.1.

TA secure loading and initialization

The TEE performs the verification of the TA authenticity and integrity based on the signature verification of the TA image. The TA image signature and the public key are included into the TA package to be loaded from the REE into the TEE. A PKI solution is used for the management of the keys involved in the TA image signing process.

In case the TA signature verification fails, the TA instantiation process is stopped.

Fulfils the SFRs: FCS_COP.1/Auth, FAU_ARP.1, FPT_FLS.1, FPT_TEE.1, FCS_COP.1/Install, FCS_CKM.1/Install, FCS_CKM.4/Install.

TA execution secure management

This function allows managing the TAs enforcing the following security operations:

- TA instance creation: TA is bound to the TEE with the UUID defined when installing the TA;
- TA session creation: TA client is bound to TA session with its UUID;
- TA isolation: TA instance is mapped to a dedicated memory space controlled in access by EL3/EPBL which configures the Memory Protection Unit, itself allocating a piece of memory to host the TEE and TAs.
- TA instance and TA session closing: all TA runtime objects are erased.

In case an authenticity or consistency error is detected on the TA code or data (cf. Trusted Storage related security services), then the execution of the TA instance is aborted.

A session data structure is created for each TA session. The CA identity is defined as a 256 byte ASCI string which is an element in the session data structure.

Fulfils the SFRs: FIA_ATD.1, FIA_UID.2, FIA_USB.1, FMT_SMR.1, FDP_RIP.1/Runtime, FAU_ARP.1, FDP_SDI.2, FPT_FLS.1, FPT_TEE.1.

TEE and TA code monitoring

The integrity of TEE code and data binary in storage is provided by firmware downgrade protection. The integrity of TA code and data binary in storage is protected by RPMB protection. The TA

identification including version will be stored and managed in RPMB. Any attempts to rollback TEE and TA code in storage will be rejected.

Fulfils the SFRs: FAU_ARP.1, FPT_FLS.1, FPT_INI.1.

TA keys access control

Access control to TA transient keys belongs to TA's own virtual address space (isolation between TAs, O.TA_ISOLATION) with instance of TA code, service libraries and runtime dynamic data. Consequently, access to TA transient keys is granted to the TA that owns the key only.

TA keys and TA runtime data are stored together in the same address space as TA instance. The only way to operate on key objects is by the use of GP Internal API.

TA keys are generated by calling the TEE APIs.

Fulfils the SFRs: FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys, FMT_SMF.1.

Trusted Storage confidentiality

The TEE implements Secure Data Objects which are structures encrypted with a derived TA key aiming at protecting data to be stored outside of the Secure World.

Fulfils the SFRs: FCS_COP.1/Store, FCS_CKM.1/Store, FCS_CKM.4/Store, FDP_ITT.1/Trusted Storage.

Trusted Storage access control

The Secure Data Objects (see definition above) are bind to a TA by TA's UUID and object ID. In addition, the TOE enforces the TA Keys Access Control SFP based on OB.TA_KEY usage.

Fulfils the SFRs: FMT_SMF.1, FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage.

Trusted Storage atomicity

This function ensures that data storage operation in trusted storage partition is atomic: upon the receipt of writing operation, a transaction process logs both previous and new object handler. If any failure occurs on commit writing operation, the transaction is rolled-back and the previous object is restored. No unsuccessful operation will not leave any partial writing.

Fulfils the SFRs: FDP_ROL.1/Trusted Storage, FDP_SDI.2.

Trusted Storage consistency and integrity

This function ensures that data storage operation in trusted storage partition is protected against unexpected modifications at runtime based on two measures:

- The persistent time is based on an REE-controlled real-time clock and on the TEE Trusted Storage for the storage of origins;
- Trusted Storage metadata prevents the loading of an overwritten version of secure object. On read/write operation, metadata is loaded for each trusted storage data. On every API access, the version in metadata and the one in storage is compared. Any mismatch will reject any further operations of trusted storage.

Fulfils the SFRs: FCS_COP.1/Store, FCS_CKM.1/Store, FCS_CKM.4/Store, FDP_SDI.2, FDP_ROL.1/Trusted Storage.

Secure Cryptographic Services

The TEE implements cryptographic services as specified in [TEE CORE 1.0] and [TEE CORE 1.1].

Fulfils the SFRs: FCS_COP.1/Digest, FCS_COP.1/Cipher, FCS_COP.1/MAC, FCS_COP.1/AE, FCS_COP.1/Enc, FCS_COP.1/Sig, FCS_COP.1/KEA.

Firmware downgrade prevention

The firmware version is stored in One Time Programmable (OTP) memory of the TOE. The verification is performed by the EL3/EPBL module at each booting of a device. It only allows the same or newer version software to continue booting.

Fulfils the SFRs: FPT_INI.1.

Trusted Service Payload Dispatcher

This function ensures the dispatching of the commands from CAs and manages the interruptions. In case of invalid, bad-formed or unexpected command, the dispatcher returns an error.

Fulfils the SFRs: FPT_FLS.1.

TEE Identification

A unique identifier is embedded into TEE binary generated at TEE integration. This unique identifier is used to bind the identification to the TEE. This identifier is visible from the TA developer.

Fulfils the SFRs: FAU_SAR.1, FAU_STG.1.

Random Numbers Generation

The TEE implements a DBRNG based on a hardware cryptographic engine accessible from the Secure World only.

Fulfils the SFRs: FCS_RNG.1.

Secure Timer

The TEE implements a Secure Timer that provides reliable time stamps, even over TEE reset. Moreover, these time stamps are monotonic between two TA “time setting” operations.

Fulfils the SFRs: FPT_STM.1/Instance time, FPT_STM.1/Persistent time, FMT_MTD.1/Persistent Time, FMT_SMF.1/Persistent Time.

TEE Time and Rollback

The TEE ensures integrity of the persistent data, even after a TEE reset. Integrity errors are detected and secure stated is preserved.

Fulfils the SFRs: FDP_SDI.2/Rollback, FPT_FLS.1/Rollback

TUI management

The TEE offers a Trusted User Interface to allow management of the peripherals. Access to peripheral is controlled such that only one TA can access it at a time and communications between the TA and the peripherals are protected in integrity and confidentiality.

Residual data is deallocated to preserve confidentiality of peripheral usage.

Fulfils the SFRs: FDP_ACC.1/TUI, FDP_ACF.1/TUI, FDP_RIP.1/TUI, FMT_MSA.1/TUI, FMT_MSA.2/TUI, FMT_MSA.3/TUI, FPT_FLS.1/TUI, FTP_TRP.1/TUI