ExynosTEE

Security Target

Revision 1.1

December 2025

ExynosTEE

SAMSUNG ELECTRONICS

SAMSUNG

THIS PAGE HAS BEEN INTENTIONALLY LEFT BLANK

Important Notice

Samsung Electronics Co. Ltd. (“Samsung”) reserves the right to make changes to the information in this publication at any time without prior notice. All information provided is for reference purpose only. Samsung assumes no responsibility for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

This publication on its own does not convey any license, either express or implied, relating to any Samsung and/or third-party products, under the intellectual property rights of Samsung and/or any third parties.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

Customers are responsible for their own products and applications. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use in applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could reasonably be expected to create a situation where personal injury or death may occur. Customers acknowledge and agree that they are solely responsible to meet all other legal and regulatory requirements regarding their applications using Samsung products notwithstanding any information provided in this publication. Customer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim (including but not limited to personal injury or death) that may be associated with such unintended, unauthorized and/or illegal use.

Samsung Electronics Co., Ltd.
1-1,Samsungjeonja-ro,Hwaseong-si,
Gyeonggi-do 18448 Korea

Home Page: *http://www.samsungsemi.com*

Trademarks

All brand names, trademarks and registered trademarks belong to their respective owners.

- Exynos, FlexOneNAND, and OneNAND are trademarks of Samsung Electronics.
- ARM, Jazelle, TrustZone, and Thumb are registered trademarks of ARM Limited.
- Cortex, ETM, ETB, Coresight, ISA, and Neon are trademarks of ARM Limited.
- Java is a trademark of Sun Microsystems, Inc.
- SD is a registered trademark of Toshiba Corporation.
- MMC and eMMC are trademarks of MultiMediaCard Association.
- JTAG is a registered trademark of JTAG Technologies, Inc.
- Synopsys is a registered trademark of Synopsys, Inc.
- I2S is a trademark of Phillips Electronics.
- I2C is a trademark of Phillips Semiconductor Corp.
- MIPI and Slimbus are registered trademarks of the Mobile Industry Processor Interface (MIPI) Alliance.

All other trademarks used in this publication are the property of their respective owners.

Revision History

Revision No.	Date	Description	Author(s)
1.0	2025/11/19	Initial release	Samsung
1.1	2025/12/05	Update the version in the reference documents	Samsung

Table of Contents

List of Figures

Figure Number	Title	Page Number

List of Tables

Table Number | **Title** | **Page Number**

List of Terms

Terms	Descriptions
Normal World	An execution environment that normal applications and OS are operating. Applications/OS in the Normal World cannot access to the resources in the Secure World
Secure World	An execution environment that trusted applications and kernel are operating. Resources in the Secure World are protected with the help of ARM TrustZone technology.

List of Acronyms

Acronyms	Descriptions
API	Application Programming Interface
BL	Boot Loader
BSP	Board Support Package, the layer of software containing hardware-specific boot firmware and device drivers.
CA	In the context of the REE/TEE system: Client Application, a program operating in the non-secure world. In the context of a PKI structure: Certification Authority
CC	Common Criteria
CM	Configuration Management
DAC	Discretionary Access Control
DAC	Dynamic Address Controller
EAL	Evaluation Assurance Level
EL3M	Execution Level 3 Monitor
ELF	Executable Linkable Format
ELx	Execution Level x
EPBL	Early Primitive Boot Loader
GID	Group Identifier
GP	GlobalPlatform
GUI	Graphic User Interface
IPC	Inter Process Communications
IV	Initialization Vector
KLDR	Kernel Loader
LK	Linux Kernel
MMU	Memory Management Unit
NWd	Normal World
OSP	Organisational Security Policy
OTP	One-Time Programmable
PKI	Public Key Infrastructure
PP	Protection Profile
PTRNG	Physical True Random Number Generator
REE	Rich Execution Environment / Regular execution environment. Also called "Normal World, Non-secure World".

RFS	ROM File System
RNG	Random Number Generator
RoT	Root of Trust
RPMB	Replay Protected Memory Block
SAR	Security Assurance Requirement
SCM	Secure Channel Manager
Secure FW	Secure Firmware
Secure OS	Secure Operating System
SWd	Secure World
SFP	Security Function Policy
SFR	Security Functional Requirement
SLSI	System Large-scale integration
SMC	Secure Monitor Call
SPD	Security Problem Definition
ST	Security Target
SVC	Supervisor Call
TA	Trusted Application
TEE	Trusted Execution Environment. Also called "Secure World".
TEK	TA Encryption Key
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSS	TOE Summary Specification
TZASC	TrustZone Address Space Controller
TZPC	TrustZone Protection Controller
UFS	Universal Flash Storage
UI	User Interface
UID	User Identifier
VFS	Virtual File System

List of References

Reference	Title
[ACM]	SOG-IS Crypto Evaluation Scheme. Agreed Cryptographic Mechanisms v1.3 Feb. 2023
[C89]	ANSI X3.159-1989 "Programming Language C.
[C99]	ISO/IEC 9899:1999 Programming languages — C
[CAPI ERR]	TEE Client API Specification v1.0 Errata and Precisions. 2.0. GlobalPlatform. Apr. 2014
[CAPI]	TEE Client API Specification, GlobalPlatform (Version 1.0, July 2010, ref: GPD_SPE_007).
[CC31R5CEM]	Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017.
[CC31R5P1]	Common Criteria, Part 1: Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, Version 3.1, Revision 5, April 2017.
[CC31R5P2]	Common Criteria, Part 2: Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components, Version 3.1, Revision, April 2017.
[CC31R5P3]	Common Criteria, Part 3: Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components, Version 3.1, Revision 5, April 2017.
[GP TEE PP]	TEE Protection Profile, GlobalPlatform (Version 1.2.1, November 2016, ref: GPD_SPE_021).
[IAPI]	Globalplatform TEE Internal Core API Specification, GlobalPlatform (Version v.1.2.1, May 2019, ref: GPD_SPE_010).
[KS2023]	A proposal for Functionality classes for random number generators (Version 2.0, 18 September 2011, part of AIS 20 / 31).
[NSI#9]	NSCIB Scheme Instruction 09, Clarification of the TSFI concept, v1.1, October 1, 2017
[OPE]	ExynosTEE Operational guidance 1.0. Samsung. Dec 2025.
[PRE]	ExynosTEE Preparative guidance 1.0. Samsung. Dec. 2025.
[RFC4122]	A Universally Unique IDentifier (UUID) URN Namespace. July 2005.

[ROT DEF]	Root of Trust Definitions and Requirements. 1.1. GlobalPlatform. July 2018.
[SA]	TEE System Architecture, GlobalPlatform (Version 1.1, January 2017, ref: GPD_SPE_009).
[SMC]	ExynosTEE 2.0 Secure Monitor Calls. 0.2. Samsung. January 2024
[T3API]	ExynosTEE 2.0 Samsung Internal Core API Extensions. 0.2. Samsung. January 2024
[USD]	ExynosTEE 2.0 User-Space Drivers. 1.3. Samsung. Sept 2023
[FSP]	Exynos TEE Functional Specification 0.4 Samsung. May 2025
[NSRPC]	Samsung ExynosTEE NSRPC Calls_0.1. Samsung. April 2025

List of Definitions

Term	Definition
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.
Monotonicity	Monotonicity is the property of a variable whose value is either always increasing or always decreasing over time

Introduction to document

Abstract

This section defines the purpose of this document. It contains the following sections:

- About This Document
- Scope Of This Document
- Intended Audience

About this document

This document contains the Security Target evaluation of ExynosTEE. This document is used for the Common Criteria evaluation of the ExynosTEE as the Target of Evaluation (TOE).

Scope of this document

This document addresses the requirements of the ASE class for Common Criteria.

Intended Audience

The intended audience of this document is as follows:

- The CC evaluator for the ExynosTEE.
- Customers that consider to use ExynosTEE as a secure OS for TEE (ARM TrustZone).
- Developers who are planning to develop TA (Trusted Application) based on ExynosTEE.

1 ST Introduction

1.1 ST Reference

Title: Samsung ExynosTEE Security Target

Revision number: 1.1

Revision date: Dec 2025

Author: Samsung Electronics Inc.

1.2 TOE Reference

TOE name: ExynosTEE

TOE version: 2.0

TOE developer: Samsung Electronics Inc.

1.3 TOE Overview

1.3.1 Introduction

The TOE is a Trusted OS as part of a Trusted Execution Environment (TEE) for embedded devices implementing GlobalPlatform TEE specifications (TEE System Architecture [SA], TEE Internal Core API [IAPI] and TEE Client API [CAPI]) as depicted in the following figure:

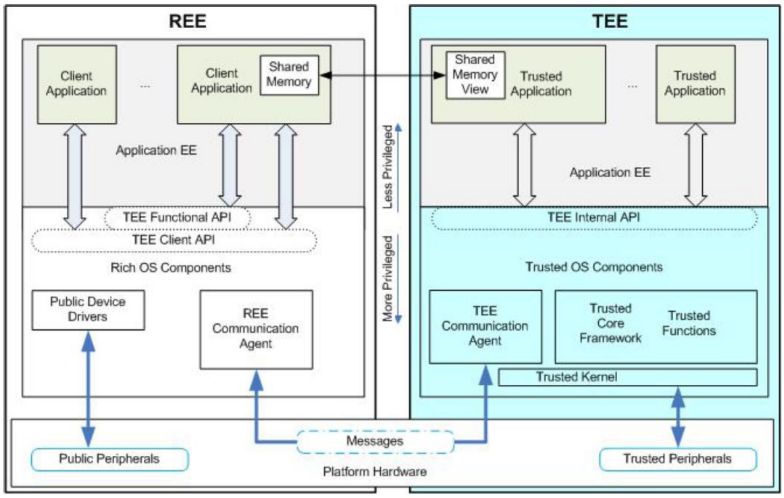

Figure 1 TEE Overall software architecture

Figure 2 below describes the TOE boundary in relation to a full TEE solution.

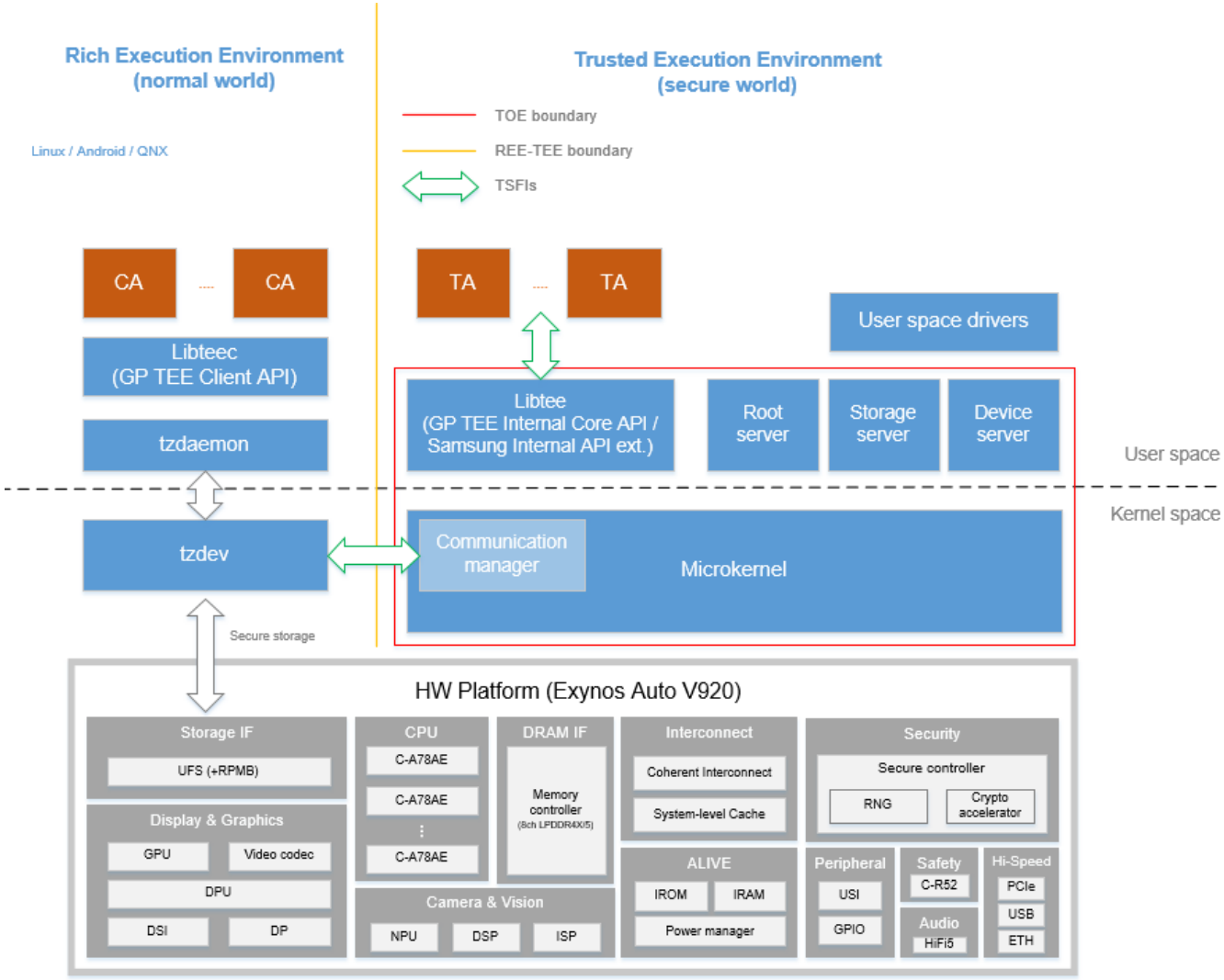

Figure 2 TOE overview

As for the major security features, the TOE supports the implementation of an execution environment isolated from any other execution environment, including the usual Rich Execution Environment (REE), and their applications. Once integrated into a TEE, the TOE gives hosting for a set of Trusted Applications (TA) and provides them with a comprehensive set of security services including: integrity of execution, secure communication with the Client Applications (CA) running in the REE, trusted storage, key management and cryptographic algorithms, time management and arithmetical API.

The TOE comprises:

- The trusted OS part of a TEE solution
- The guidance for the secure usage of the TEE OS functionality after delivery.
- The guidance for the integration of the TOE into a TEE solution.

The TOE does not comprise:

- The Trusted Applications
- The Rich Execution Environment
- The Client Applications
- The underlying platform (hardware and firmware)

1.3.2 TOE Type

The TOE type is a Trusted OS, which is part of a Trusted Execution Environment (TEE) for embedded devices implementing GlobalPlatform TEE specifications (TEE System Architecture [SA], TEE Internal Core API [IAPI] and TEE Client API [CAPI]) as depicted in Figure 1. Figure 2 describes the TOE boundary in relation to a full TEE solution.

1.3.3 TOE Usage & major security features

1.3.3.1 TEE-based functionality

The security functionality in operational status which is in the scope of the evaluation is described in section 1.4.2.

This security functionality defines the logical boundary of the TOE, while the security functionality provided by the Trusted Applications is out of the scope of the TOE.

1.3.3.2 TOE usage

The TOE purpose is to securely host and execute TAs (Trusted Applications), enforcing their mutual isolation and isolation from other execution environments, and ensuring integrity and/or confidentially of the assets managed.

1.3.4 Non-TOE Hardware/Software/Firmware

The TOE requires the following non-TOE hardware, firmware, and software for operation:

- Samsung Automotive SoC Exynos Auto V9 (S5AHR80A) which has the features:
    - ARM v8 based 64-bit application processor (aarch64) SoC
    - Trustzone enabled
    - RAM, which stores the runtime data of ExynosTEE and REE components
    - ROM, which keeps secure boot
    - Non-volatile storage, which keeps ExynosTEE and REE components (e.g. Linux, Android, QnX)
    - Secure OTP memory, which contains sensitive data required by the secure bootloader
    - Secure bootloader enabling secure boot (Samsung's LK bootloader, version ≥ 6).
- Required peripherals to operate the host device, such as display, touch screen or others.
- Implementation of the REE side (Linux, Android or QnX) including the following:
    - Any implementation of GP TEE Client API as libtee
    - REE kernel module for SMC execution as “tzdev”
    - Any implementation of the user-space wrapper “tzdaemon”
- Trusted Applications (TAs) and Client Applications (CAs)
- User-space drivers for the TEE side.

- "Loadable firmware" that provides support in the execution to the user-space drivers.
- Random Number Generator
    - TRNG(True Random Number Generator) generates random numbers from an amplified and sampled thermal noise of a chip.
    - The generated random numbers are non-deterministic and unpredictable bit stream.

Note: The underline platform has been certified…

1.4 TOE description

1.4.1 TOE physical scope

The TOE consists of the set of software files and guidance documents that are delivered in electronic format through a file transfer. The following table lists the guidance documents and their format.

Title	Version	Format
ExynosTEE 2.0	2.0.0	Binary
ExynosTEE Preparative guidance	1.0	PDF
ExynosTEE Operational guidance	1.0	PDF
GlobaPlatform TEE Internal Core API specification	1.2.1	PDF
ExynosTEE 2.0 Samsung Internal Core API Extensions	0.2	PDF
ExynosTEE 2.0 Secure Monitor Calls	0.2	PDF

Table 1 TOE Deliverables

1.4.2 TOE logical scope

The logical scope of the TOE can be divided in the following categories:

- 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
- Trusted storage of TA and TEE data and keys, ensuring consistency, confidentiality, atomicity and binding to the TEE
- Cryptographic API including:
    - Generation of cryptographic keys
    - Support for cryptographic algorithms as described in Table 2. The table includes only recommended and legacy mechanisms since [ACM]. For the second, special notes have been added. Other mechanisms not in the Table 2 are not in the scope of the evaluation.
- TA instantiation that ensures the authenticity and contributes to the integrity of the TA code
- Correct execution of TA services; note that [GP TEE PP] lists more security features, but the TOE relies on the underlying platform to provide these features.

Title	Cryptographic algorithm	Supported key sizes	Corresponding standards
Symmetric Cipher	AES (CBC[1], CTR[1], XTS[2], CCM, GCM)	128, 192, 256	FIPS 197 (AES) NIST SP800-38A (CBC, CTR) IEEE Std 1619-2007 (XTS) RFC 3610 (CCM) NIST 800-38D (GCM)
	TDES[3] (CBC)	112, 168	FIPS 46 (TDES) FIPS 81 (CBC)
Digest	SHA224[3], SHA256, SHA384, SHA512	Not applicable	FIPS 180-4 (SHA224, SHA256, SHA384, SHA512)
MAC	AES (CMAC, CBC MAC)	128, 192, 256	NIST SP800-38B (CMAC) ISO9797 (CBC MAC)
	TDES[3] (CBC MAC)	112, 168	ISO9797 RFC1423
	HMAC (SHA1[3], SHA224[3], SHA256, SHA384, SHA512)	Limited by the block size of the hash function	RFC 2202 (SHA1) RFC 4231 (SHA224, SHA256, SHA384, SHA512)
Asymmetric Cipher	RSAES (PKCS#1 v1.5[3], OAEP)	2048[3], 3072	PKCS#1
Digital Signature	RSASSA (PKCS#1 v1.5[3], PSS)	2048[3], 3072	PKCS#1
	DSA	TEE_ALG_DSA_SHA224 : 2048[3] bits TEE_ALG_DSA_SHA256 : 2048[3] or 3072 bits	FIPS 186-4
	ECDSA[4]	256, 384, 521	FIPS 186-4 ANSI X9.62

[1] Only approved if used together with an additional integrity-check mechanism. Otherwise, legacy (check [ACM]).

[2] XTS only supports key sizes 128 and 256 bits

[3] Legacy. Please check [ACM].

[4] NIST-P curves are used.

Key Exchange (Shared secret derivation)	DH	2048[3] bits, multiple of 8 bits.	PKCS #3
	ECDH[4]	256, 384, 521	NIST SP800-56A

Table 2 Supported cryptographic algorithms

Note: TEE_ALG_DSA_SHA1 is supported (corresponding to FIPS 186-4 standard) however it is not to be used under the scope of this evaluation as it is not recommended.

1.4.3 Reference device life cycle

The TOE is created using a simplified implementation of the V-Model as the chosen method of development; the V-Model is a graphical representation of a systems development lifecycle that functions well with software development projects, primarily due to each phase of the model defining specific deliverables and including a review process.

Figure 2 below depicts the various stages product development within the life cycle of the TOE, see [ALC] for further information on the life cycle:

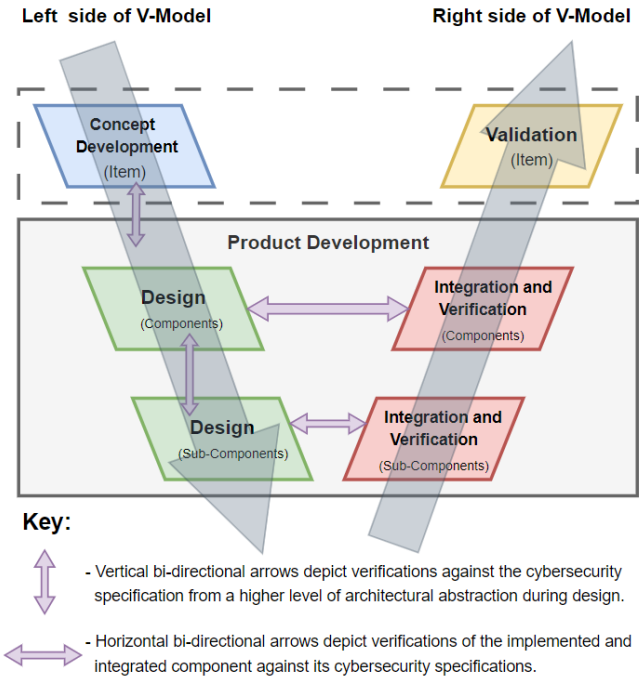

Figure 3 Stages of the Lifecycle (V-Model)

The five stages of the lifecycle are as follows:

- **Concept development:**
  - The concept development stage identifies the cybersecurity requirements for the lifecycle of the TOE through threats extracted from the TARA report generated during this stage.

- **Design:**
    - The design stage uses the cybersecurity requirements and controls defined in the Concept development stage to analyze the risks, so that the design for the product can be updated in accordance with these risks.
- **Integration and Verification:**
    - The integration and verification stage implements the design into the product, as well as verifying that the previously identified risks have been mitigated through the implementation of this design.
- **Validation:**
    - The validation stage uses all artifacts generated from the previous stages, from Concept (incl. TARA result) to Verification Results, in a process to validate the goals and requirements.
- **Maintenance:**
    - The maintenance stage describes the activities that follow after the completion and release of the product to customers for use, this is also known as the operational phase whereby the TOE is operating in a customer's environment.

2 Conformance claims

2.1 CC Conformance claim

This Security Target claims to be conformant to the Common Criteria version 3.1, Revision 5.

Furthermore, it claims to be conformant to the CC Part 2 [CC31R5P2] and CC Part 3 [CC31R5P3] extended.

This Security Target has been built with the Common Criteria for Information Technology Security Evaluation; Version 3.1 which comprises:

- [CC31R5P1]
- [CC31R5P2]
- [CC31R5P3]
- [CC31R5CEM]

2.2 PP claim

This Security Target does not claim conformance to any Protection Profile.

2.3 Package claim

This Security Target claims conformance to the assurance package EAL3 augmented with ALC_FLR.1, as well as AVA_VAN.2 augmented with the AVA_TEE.2 package taken from [GP TEE PP].

2.4 Conformance claim rationale

While this Security Target does not claim conformance to a protection profile, it uses the particular version of [GP TEE PP] as a model of presentation to define the security functionality.

The protection profile augments AVA_VAN.2 with the profile defined package AVA_TEE.2; as such, this extended component has been chosen to augment AVA_VAN.2 to provide a sufficient level of assurance for this TOE type.

The Security Problem Definition and the Security Objectives are based in the ones described in the identified version of [GP TEE PP].

The following table shows the relation between the identified version of [GP TEE PP] and this ST in terms of the Security Problem Definition:

SPD element	[GP TEE PP]	This ST
T.ABUSE_FUNCT	x	x
T.CLONE	x	x
T.FLASH_DUMP	x	X (refined to exclude TEE data)
T.IMPERSONATION	x	x
T.ROGUE_CODE_EXECUTION	x	x
T.PERTURBATION	x	x
T.RAM	x	x
T.RNG	x	x
T.SPY	x	x
T.TEE_FIRMWARE_DOWNGRADE	x	x
T.STORAGE_CORRUPTION	x	x
OSP.INTEGRATION_CONFIGURATION	x	x (extended to include integration of TOE into a full TEE solution)
OSP.SECRETS	x	x
A.PROTECTION_AFTER_DELIVERY	x	x
A.ROLLBACK	x	x
A.TA_DEVELOPMENT	x	x
A.SECURE_INITIALIZATION		x
A.CONNECT		x
A.CONFIGURATION		x
A.CAR_PHYSICAL_PROTECTION		x
A.RPMB		x

Table 3 Relation of SPD in [GP TEE PP] and this ST

The following table shows the relation between the identified version of [GP TEE PP] and this ST in terms of the Security Objectives:

Security objective	[GP TEE PP]	This ST
O.CA_TA_IDENTIFICATION	x	x
O.INITIALIZATION		This objective has been shifted to OE.INITIALIZATION
O.KEYS_USAGE	x	x
O.OPERATION	x	x
O.RNG	x	This objective has been shifted to OE.RNG
O.RUNTIME_CONFIDENTIALITY	x	x
O.RUNTIME_INTEGRITY	x	x
O.TA_AUTHENTICITY	x	x
O.TA_ISOLATION	x	x
O.TEE_DATA_PROTECTION	x	x
O.TEE_ID	x	The objective has been amended to state that the identifier must be provided by the hardware as per OE.UNIQUE_TEE_ID.
O.TEE_ISOLATION	x	x
O.TRUSTED_STORAGE	x	x
O.INSTANCE_TIME	x	This objective has been shifted to OE.INSTANCE_TIME
OE.INSTANCE_TIME		x
OE.INITIALIZATION		x
OE.INTEGRATION_CONFIGURATION	x	x
OE.PROTECTION_AFTER_DELIVERY	x	x
OE.ROLLBACK	x	x
OE.SECRETS	x	x
OE.TRUSTED_HARDWARE		x
OE.TRUSTED_FIRMWARE		x
OE.UNIQUE_TEE_ID		x
OE.RNG		x

OE.TA_DEVELOPMENT	x	x
OE.CAR_PHYSICAL_PROTECTION		x
OE.RPMB		x

Table 4 Relation of Security objectives in [GP TEE PP] and this ST

The following table shows the relation between the identified version of [GP TEE PP] and this ST in terms of the Security Functional Requirements:

SFR	**[GP TEE PP]**	**This ST**
Identification		
FIA_ATD.1	x	x
FIA_UID.2	x	x
FIA_USB.1	x	x
FMT_SMR.1	x	x
Confidentiality, Integrity and Isolation		
FDP_IFC.2/Runtime	x	x This functionality is supported by the platform as per A.TRUSTED_HARDWARE.
FDP_IFF.1/Runtime	x	
FDP_ITT.1/Runtime	x	This is provided by the hardware only and the SFR is not included.
FDP_RIP.1/Runtime	x	x
Cryptography		
FCS_COP.1	x	X This SFR is iterated twice in this ST to provide more granularity in the definition of cryptographic functionality.
FDP_ACC.1/TA_Keys	x	x
FDP_ACF.1/TA_Keys	x	x
FMT_MSA.1/TA_Keys	x	x
FMT_MSA.3/TA_Keys	x	x
Initialization, Operation and Firmware Integrity		
FAU_ARP.1	x	x This is also supported by the hardware/firmware.
FPT_FLS.1	x	

FDP_SDI.2	x	x
FPT_INI.1	x	The hardware and firmware (operational environment) have to perform the checks described in [GP TEE PP].
FMT_SMF.1	x	x
FPT_TEE.1	x	x
TEE identification		
FAU_SAR.1	x	The TOE only provides the unique identifier to TA users.
FAU_STG.1	x	The protection is provided by the hardware only and the SFR is not included.
Instance time		
FPT_STM.1/Instance time	x	Associated objective is shifted to the environment and the SFR is not included.
Random number generation		
FCS_RNG.1	x	Random numbers must be provided by the hardware (operational environment).
Trusted storage		
FDP_ACC.1/Trusted Storage	x	x
FDP_ACF.1/Trusted Storage	x	x
FDP_ROL.1/Trusted Storage	x	x
FMT_MSA.1/Trusted Storage	x	x
FMT_MSA.3/Trusted Storage	x	x
FDP_ITT.1/Trusted Storage	x	x
Additional SFRs		
FCS_CKM.1/GP-API		X
FCS_CKM.4		x
FDP_ITC.1/TEK		x
FDP_ACC.1/TEK		x
FDP_ACF.1/TEK		x

Table 5 Relation of the SFRs in [GP TEE PP] and this ST

3 Security problem definition

3.1 Description of the assets

This section presents the assets of the TOE and their properties, each asset defines its properties (refer to the “List of Definitions” for the description of some of these properties).

TEE identification

TEE identification data (TEE Identifier) that is globally unique among all GlobalPlatform TEEs whatever the manufacturer, vendor or integrator. This data is typically stored in the Trusted OTP memory of the TEE.

Properties: unique and non-modifiable.

Application Note:

The TEE identifier is intended to be public and exposed to any software running on the device, not only to Trusted Applications. For this TOE, the TEE identifier is exposed to Trusted Applications only, and as part of the objective for the environment OE.TA_Development the user shall implement at least one TA that exposes the TEE identifier to the REE side.

RN

Random Numbers.

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 shared with REE and potentially accessible by it.

Properties: authenticity and consistency (which implies runtime integrity).

TA data and keys

Data and keys managed and stored by a TA using the TOE 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 shared with REE and potentially accessible by it.

Properties: authenticity, consistency (which implies runtime integrity), atomicity, confidentiality and device binding.

TA instance time

Monotonic time 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

Runtime TEE data, including 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 TOE.

TEE persistent data

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

Properties: authenticity, consistency (which implies runtime integrity), confidentiality and device binding.

TEE firmware

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

Properties: authenticity, integrity.

TEE initialization code and data

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

Properties: integrity.

Application Note:

After power-up, the secure bootloader of the hardware platform checks the integrity and authenticity of the TOE image (OE.INITALIZATION). This also includes verification of authenticity and consistency of TEE data, which is part of the TOE image. During that process, a failure during initialization of a particular software or hardware component (service, device driver, device binding failure) results in entering into the secure state (halt of the system).

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. Root Storage Key is to be protected to provide the TEE with the device binding, authenticity and confidentiality of the secure storage.

Properties: integrity and confidentiality.

Application Note:

The TEE storage root of trust contains the asset Storage Root of Trust key; a key derived from OTP memory stored keys.

3.2 Users / Subjects

There are two kinds of users of the TOE (as also defined in [GP TEE PP]): Trusted Applications, which use the TOE services through the TEE Internal Core API, and the Rich Execution Environment, which uses the TOE's services exported by the Trusted Applications.

Trusted Application (TA)

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

Rich Execution Environment (REE)

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

3.3 Threats

This ST targets threats to the TOE assets that arise during the end-usage phase and can be achieved by software means. Attackers are individuals or organizations with remote or physical (local) access to the automotive component embedding the TEE. Users with physical access to the car and the car key are not considered potential attackers.

The possible threats can be classified into the following categories:

- Attacks through either wireless or standard physical connection of other equipment, to the vehicle systems.
- Attacks from vehicle system software outside of the TEE.

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

The threats considered, and, therefore, mitigated by the TOE presented in this document are threats which may materialize through the interfaces available to the REE and/or TAs.

T.ABUSE_FUNCT

An attacker accesses TEE functionalities 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 initialization code and data (integrity), TEE runtime data (confidentiality, integrity), RN (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, for instance, in using commands in unexpected contexts or with unexpected parameters, impersonation of authorized entities or exploiting badly implemented reset functionalities that provides undue privileges. In particular a fake application running in the Rich OS 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 threat is not countered by the TEE alone and must be taken into account in the design of the use case, 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 of a first device on a second device and makes this device accept them as genuine data.

Assets threatened directly: All data and keys (authenticity, device-binding), 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 data and potentially allowing the attacker to mount other attacks.

Assets threatened directly (confidentiality, authenticity, consistency): TA data and keys.

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. Another example consists of unsoldering a flash memory and dumping its content, revealing a secret key that provides privileged access to many devices of the same model.

T.IMPERSONATION

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

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

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

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, RN.

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

Application Note:

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

T.PERTURBATION

An attacker modifies the behavior of the TEE or 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 initialization code and data (integrity), TEE storage root of trust (confidentiality, integrity), TEE runtime data (confidentiality, integrity), RN (confidentiality, integrity).

Assets threatened indirectly: All data and keys (confidentiality, authenticity, consistency) 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 exploit e.g. multi-threading or 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 initialization code and data.

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

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

Application Note:

When the REE and the TEE have shared memory, an attack path consists in the (partial) memory dump (read/write) by the REE. During the identification phase, another example of attack path is to snoop on a memory bus, revealing code that is only decrypted at run-time, and finding a flaw in that code that can be exploited.

T.RNG

An attacker obtains information in an unauthorized manner about random numbers used by the TOE. This may occur for instance by a lack of entropy of the random numbers generated by the product, or because the attacker forces the output of a partially or totally predefined value. Loss of unpredictability (the main property of random numbers) is a problem in case they are used to generate cryptographic keys. Malfunctions or premature ageing may also allow getting information about random numbers.

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

T.SPY

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

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

Application Note:

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

T.TEE_FIRMWARE_DOWNGRADE

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

Assets threatened directly (integrity): TEE firmware.

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

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 behavior 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), TEE firmware (authenticity, integrity), TA data and keys (confidentiality, authenticity, consistency), TA instance time (integrity), TA code (authenticity, consistency).

Application Note:

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

3.4 Organizational security policies

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

OSP.INTEGRATION_CONFIGURATION

Integration and configuration of the TEE by the device manufacturer shall rely on guidelines defined by the TEE provider, which fulfill the requirements set in GlobalPlatform TEE specifications and state all the security requirements for the device manufacturer issued from the TOE evaluation.

Application Note:

In this ST, this OSP includes the integration of the TOE into the TEE solution as the TOE is just a subset of the TEE.

OSP.SECRETS

Generation, storage, distribution, destruction, injection of secret data in the TEE or any other operation performed outside the TEE shall enforce integrity and confidentiality of these data. This applies to secret data injected before 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, confidential data).

3.5 Assumptions

This section states the assumptions that hold on the TOE operational environment.

A.PROTECTION_AFTER_DELIVERY

It is assumed that the TOE is protected by the environment after delivery and before entering the final usage phase. It is assumed that the persons manipulating the TOE in the operational environment apply the TOE guidelines (e.g. user and administrator guidance, installation documentation, personalization guide). It is also assumed that the persons responsible for the application of the procedures contained in the guides, and the persons involved in delivery and protection of the product have the required skills and are aware of the security issues.

Application Note:

The certificate is valid only when the guidelines that are listed in the guidance documents described in the physical scope are followed.

A.ROLLBACK

It is assumed that TA developers do not rely on protection of TEE persistent data, TA data and keys and TA code against full rollback.

A.RPMB

The hardware platform includes the RPMB, which is a secure and reliable storage area provided by the UFS storage device. RPMB is considered a secure storage solution for the following reasons:

Physical Security: RPMB is located within the flash memory chip, providing physical chip-level security. It ensures protection against external physical intrusion or attacks, safeguarding the data stored within.

Data Integrity: RPMB guarantees data integrity. It uses HMAC (Hash-based Message Authentication Code) to authenticate stored data, ensuring that it has not been tampered with or altered. This allows verification of data against any unauthorized modifications.

Access Control: RPMB implements access control, allowing only authenticated devices to access the data. This prevents unauthorized access from external sources and maintains the security of the stored data. A counter is used each time the data is written in RPMB, therefore, replay attacks are countered and freshness of the data is

ensured.

Additionally, the storage server in the ExynosTEE encrypts the TEE persistent data (secure objects) to be stored in the RPMB with unique symmetric key for each TA. As a result, the secure object stored in the RPMB is isolated because it cannot be accessed from other TAs.

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.

A.SECURE_INITIALISATION

It is assumed that the non-TOE hardware and firmware required by the TOE guarantee the secure initialization and protection of the TOE, until the TOE has been initialized, as mandated by [GP TEE PP] including:

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

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

A.CONFIGURATION

It is assumed that the TOE will be properly configured and installed on the appropriate, dedicated hardware. The set of software packages forming the TOE must be installed during installation time in accordance with the installation instructions provided in the installation guidance document.

A.CAR_PHYSICAL_PROTECTION

It is assumed that:

- The TOE operational environment isolates the hardware from physical probing and manipulation.
- The car user(s) (those who have legitimate physical access to car and its parts) are trusted. They will not manipulate or modify the TOE, either willfully or unwilfully.

A.CONNECT

This assumption addresses security concerns related to the manipulation of the TOE through its legitimate interfaces. Internal communication paths to interface points such as peripheral devices are assumed to be adequately protected.

Application Note:

It is assumed that the SoC protects the communication of internal (TA and ExynosTEE) data that are transmitted in clear to physically separated peripherals to ensure the integrity and confidentiality of the data transmitted.

A.PEER.FUNC

It is assumed that the non-TOE hardware, firmware and software that are required for the TSF operation behave as expected.

Application note:

This concerns for instance, secure boot function, CPU exception levels, memory management unit, TrustZone isolation integration, OTP memory and RPMB integration.

A.PEER.MGT

It is assumed that the non-TOE hardware, firmware and software that are required for the TSF operation are configured and managed to provide the expected security services to the TSF.

4 Security objectives

4.1 Security objectives for the TOE

This section states the security objectives for the TOE.

O.CA_TA_IDENTIFICATION

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

Application Note:

Client properties (both CA and TA) are managed either by the Rich 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 TOE resident TAs MUST always be determined by the TOE and the determination of whether it is a TA or not MUST be as trustworthy as the TOE itself
- When the Client identity corresponds to a TA, then the TOE 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 TOE
- When the Client identity does not correspond to a TA, then the Rich OS is responsible for ensuring that the Client Application cannot tamper with its own properties. However, this information is not trusted by the TOE.

O.KEYS_USAGE

The TOE shall enforce on cryptographic keys the usage restrictions set by their creators (TA identifier).

O.TEE_ID

The TOE shall provide means to a TA to retrieve the unique TEE identifier provided by the hardware.

Application Note:

The hardware shall provide the unique identifier as per OE.UNIQUE_TEE_ID.

O.OPERATION

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

- Protect itself against abnormal situations caused by programmer errors or violation of good practices by the REE (and the CAs indirectly) or by the TAs

- Control the access to its services by the REE and TAs: The TOE shall check the validity of any operation requested from either the REE or a TA, at any entry point into the ToE

- 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. The REE may be harmful but «the implementation (TEE) still guarantees the stability and security of TEE » (refer to [CAPI]). In any case, a Trusted Application MUST NOT be able to use a programmer error on purpose to circumvent the security boundaries enforced by an implementation (refer to [IAPI] and [SA])
- Entry points (refer to [SA]): Software in the REE must not be able to call directly to TOE Functions or Trusted Core Framework. The REE software must go through protocols such that the TOE or Trusted Application performs the verification of the acceptability of the operation that the REE software has requested.

O.RUNTIME_CONFIDENTIALITY

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

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

O.RUNTIME_INTEGRITY

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

O.TA_AUTHENTICITY

The TOE shall verify code authenticity of Trusted Applications and maintain the confidentiality of TA code when stored in non volatile memory.

Application Note:

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

O.TA_ISOLATION

The TOE 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 the integrity of the TEE.

O.TEE_DATA_PROTECTION

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

O.TEE_ISOLATION

The TOE shall prevent the REE and the TAs from accessing the TEE 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 TEE resources during their usage and REE/TAs.

O.TRUSTED_STORAGE

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

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

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

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

Asset	In non-volatile memory	In volatile memory
TEE firmware	OE.INITIALIZATION	O.RUNTIME_INTEGRITY
TEE runtime data	N/A	O.RUNTIME_INTEGRITY, O.RUNTIME_CONFIDENTIALITY
TA code	O.TA_AUTHENTICITY	O.RUNTIME_INTEGRITY
TA data and keys	O.TRUSTED_STORAGE OE.RPMB	O.RUNTIME_INTEGRITY, O.RUNTIME_CONFIDENTIALITY
TEE persistent data	O.TEE_DATA_PROTECTION	O.TEE_DATA_PROTECTION

Table 6 Relationship between security objectives and the assets stored in each memory

Application Note:

Compared to the [GP TEE PP], the TOE requires that the hardware provides some functionality allowing the TOE to implement a full trusted storage solution, as described in OE.TRUSTED_HARDWARE and OE.RPMB.

4.2 Security objectives for the operational environment

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

OE.INTEGRATION_CONFIGURATION

Integration and configuration of the TEE by the device manufacturer shall rely on guidelines defined by the TEE provider that fulfill the requirements set in GlobalPlatform TEE specifications and state all the security requirements for the device manufacturer issued from the TOE evaluation.

Application Note:

In this ST, this OSP includes the integration of the TOE into the TEE solution as the TOE is just a subset of the TEE, therefore the integration of the TOE into the TEE shall follow the guidelines defined by TEE provider.

OE.PROTECTION_AFTER_DELIVERY

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

Application Note:

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

OE.ROLLBACK

The TA developer shall take into account that the TEE does not provide full rollback protection of TEE persistent data, TA data and keys and TA code.

OE.RPMB

The hardware platform shall include the RPMB, which is a secure and reliable storage area provided by the UFS storage device. The TEE persistent data (secure object) used by each TA shall stored in the RPMB area, and the secure object shall be secured. The TA data shall be encrypted to the TEE persistent data storage with a different encryption key for each TA. As a result, the secure object stored in the RPMB shall be isolated because it cannot be accessed from other TAs or other external entities.

OE.SECRETS

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

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 the Trusted Applications:

- CA identifiers are generated and managed by the REE, outside the scope of the TEE; TAs do not assume that CA identifiers are genuine.
- TAs do not disclose any sensitive data to the REE through any CA (interaction with the CA may require authentication means).
- The TA developer shall not use a certificate issued by the integrator with a privilege other than “PUBLIC”. Using a certificate with a different privilege is considered out of scope of this evaluation.
- TAs shall expose the TEE identifier (public) to any software running on the device
- 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.
- TA developers are expected to use the cryptographic functionality provided by the Global Platform API available in the TOE and not develop their own implementations.

OE.INITIALIZATION

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

- the integrity of the TEE initialization code and data used to load the TEE firmware;
- the authenticity of TEE persistent data. In particular, TA properties, TEE keys and all security attributes;
- the integrity and authenticity of the TEE firmware;
- the integrity and authenticity of the TOE image;
- entering a secure state upon detection of TEE initialization failure or unexpected commands;
- and that the TEE is bound to the SoC of the device. In particular, the TEE shall protect the TEE firmware against downgrade attacks;

Application Note:

The fact that the process is bound to the SoC means that the root of trust for the TEE data cannot be modified or tampered with (refer to [SA]).

OE.INSTANCE_TIME

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

OE.TRUSTED_HARDWARE

SoC Hardware and secure Firmware implements the protocols and mechanisms required by the TSF to support the enforcement of the security policy. Those systems provide the functions required by the TOE and are sufficiently protected from any attack that may cause those functions to provide false results.

In particular:

- A security mechanism enabling the TOE to isolate the REE and the TEE side, through the ARMv8 platform with TrustZone technology.
- A security mechanism enabling the TOE to isolate different internal processes, through a configurable hardware MMU and privilege levels.
- Disabling or closing the debug interfaces.

OE.TRUSTED_FIRMWARE

The secure firmware and secure drivers are trustworthy; the secure drivers and firmware shall be provided by the manufacturer, designed for the specific board purchased.

OE.UNIQUE_TEE_ID

A globally unique and non-modifiable TEE identifier that is stored in secure OTP memory. Generation of the TEE identifier, outside or inside the TEE, shall enforce the statistical uniqueness of this data.

OE.CONFIGURATION

It is assumed that the TOE will be properly configured and installed on the appropriate, dedicated hardware. The set of software packages forming the TOE must be installed during installation time in accordance with the installation instructions provided in the installation guidance document. [PRE].

OE.RNG

The hardware platform shall provide a random number generator. This RNG shall provide random numbers of sufficient quality, providing unpredictability and shall have sufficient entropy.

OE.CAR_PHYSICAL_PROTECTION

The hardware where the TOE is integrated into shall be protected against physical probing and manipulation. This can be achieved through a physical barrier such as an epoxy layer combined with enclosing the hardware components in a metal box. This kind of protection is usual for similar devices where the TOE will be embedded into, such as car ECUs.

Additionally, the TOE and its related hardware will be deployed inside a car as their working environment. It is expected that the car itself contains the usual security measures (key locks, possibly an alarm system) that would prevent up to a certain level the unnoticed manipulation or unauthorized physical access to the TOE.

4.3 Security objectives rationale

4.3.1 Rationale for the threats

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

- OE.INITIALIZATION ensure 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.
- OE.TRUSTED_HARDWARE contributes in providing logical and physical isolation between TAs, the TAs and the kernel layer and the TEE (TAs and OS) from the REE.

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

- O.TEE_ID provide the unique TEE identification means.
- OE.INITIALIZATION ensure that the TEE is bound to the SoC of the device.
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data, in particular TSF data used to

bind the TEE to the device.

- O.RUNTIME_INTEGRITY prevents against 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.
- OE.UNIQUE_TEE_ID ensures that the device ID is in fact unique.
- OE.TRUSTED_HARDWARE contributes to reducing the attack surface to perform cloning attacks by enforcing a system of privilege levels with logical and physical measures that difficult disclosing assets from unauthorized entities.

T.FLASH_DUMP

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.
- O.RUNTIME_CONFIDENTIALITY prevents against disclosure of data at runtime.

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

- OE.INITIALIZATION ensure that the TEE security functionality is correctly initialized and that the integrity of TEE firmware is verified.
- O.OPERATION ensures correct operation of the security functionality.
- O.RUNTIME_CONFIDENTIALITY covers runtime TEE data which might influence the behavior 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 behavior 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 other than the end-user phase.

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

- OE.INITIALIZATION ensure that the TEE security functionality is correctly initialized.
- OE.INSTANCE_TIME ensures the reliability of instance time stamps.
- O.OPERATION ensures correct operation of the security functionality and a proper management of failures.

- O.RUNTIME_CONFIDENTIALITY covers runtime TEE data which might influence the behavior 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 behavior of the TEE.
- O.TEE_ISOLATION enforces the separation between the TEE and the outside (REE and TAs).
- OE.CAR_PHYSICAL_PROTECTION ensures the physical protection of the hardware running the TEE while in its legitimate environment and protects against physical attacks.

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

- OE.INITIALIZATION ensure that the TEE security functionality is correctly initialized and that the initialization 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.
- OE.CAR_PHYSICAL_PROTECTION ensures the physical protection of the RAM used by the TEE while in its legitimate environment and prevents any physical attacks.
- OE.TRUSTED_HARDWARE contributes to the mitigation of the unauthorized access of volatile memory data by enforcing logical separation of memory spaces between TAs, the OS and providing isolation from REE environment or any debug interface.

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

- OE.INITIALIZATION ensure the correct initialization of the TEE security functions, in particular the RNG.
- OE.RNG ensures that random numbers are unpredictable and have sufficient entropy.
- O.RUNTIME_CONFIDENTIALITY ensures that confidential data is not disclosed.
- O.RUNTIME_INTEGRITY protects against unauthorized modification of data.
- OE.TRUSTED_HARDWARE contributes to the mitigation of the unauthorized access of random numbers or data derived from them by enforcing logical separation of memory spaces between TAs, the OS and providing isolation from REE environment or any debug interface.

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.
- OE.CAR_PHYSICAL_PROTECTION ensures the physical protection of the hardware running the TEE while in its legitimate environment and protects against physical attacks.

T.TEE_FIRMWARE_DOWNGRADE The combination of the following objectives ensures protection against TEE firmware downgrade:

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

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

- O.OPERATION ensures the correct operation of the 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.
- OE.INITIALIZATION ensure that the firmware that is executed is genuine.
- OE.ROLLBACK states the limits of the properties enforced by the TSF.
- OE.TRUSTED_HARDWARE contributes to the mitigation of unauthorized access to non-volatile storage between TAs and between REE and Tas.
- OE.RPMB ensures that persistent data of TAs is integrity and replay protected.

4.3.2 Rationale for the OSPs

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

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

4.3.3 Rationale for the assumptions

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

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

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

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

A.SECURE_INITIALIZATION The objective OE.INITIALIZATION directly covers this assumption.

A.CONFIGURATION The objective OE. CONFIGURATION directly covers this assumption by requiring proper installation and configuration for starting up the TOE in a secure state.

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

A.CONNECT The objectives OE.TRUSTED_HARDWARE and OE.TRUSTED_FIRMWARE directly cover this assumption by requiring that:

- the Hardware/Firmware or the software environment provide the functions required by the TOE, and are

sufficiently protected from any attack that may cause those functions to provide false results.

- the Hardware/Firmware or the software environment demands physical and logical protection equivalent to the TOE.

A.PEER.MGT The objectives OE.TRUSTED_HARDWARE and OE.TRUSTED_FIRMWARE directly cover this assumption by requiring that:

- the Hardware/Firmware are under the same management domain as the TOE.
- the Hardware/Firmware is managed based on the same rules and policies applicable to the TOE.

A.PEER.FUNC The objectives OE.TRUSTED_HARDWARE and OE.TRUSTED_FIRMWARE directly cover this assumption by requiring that:

- the Hardware/Firmware implement the protocols and mechanisms required by the TSF to support the enforcement of the security policy (OE.TRUSTED_HARDWARE).
- all additional Firmware/Software (such as drivers) does not interfere with the security policy enforced by the TSF (OE.TRUSTED_FIRMWARE).

4.3.4 Coverage

The following table shows the mapping of the objectives to threats, OSPs and assumptions

Threat, OSPs, Assumptions	Objectives
T.ABUSE_FUNCT	OE.INITIALIZATION, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_AUTHENTICITY, O.TEE_DATA_PROTECTION, O.TEE_ISOLATION, O.KEYS_USAGE, OE.TA_DEVELOPMENT, OE.TRUSTED_HARDWARE
T.CLONE	O.TEE_ID, OE.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, OE.UNIQUE_TEE_ID, OE.TRUSTED_HARDWARE
T.FLASH_DUMP	O.TRUSTED_STORAGE
T.IMPERSONATION	O.CA_TA_IDENTIFICATION, O.OPERATION, O.RUNTIME_INTEGRITY, O.RUNTIME_CONFIDENTIALITY
T.ROGUE_CODE_EXECUTION	OE.INITIALIZATION, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.TA_AUTHENTICITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY
T.PERTURBATION	OE.INITIALIZATION, OE.INSTANCE_TIME, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.TA_AUTHENTICITY O.RUNTIME_INTEGRITY, O.TA_ISOLATION,

	O.TEE_DATA_PROTECTION, O.TEE_ISOLATION, OE.CAR_PHYSICAL_PROTECTION
T.RAM	OE.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION, OE.CAR_PHYSICAL_PROTECTION, OE.TRUSTED_HARDWARE
T.RNG	OE.INITIALIZATION, OE.RNG, OE.TRUSTED_HARDWARE, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY
T.SPY	O.RUNTIME_CONFIDENTIALITY, O.TA_ISOLATION, O.TEE_ISOLATION, O.TRUSTED_STORAGE, OE.CAR_PHYSICAL_PROTECTION
T.TEE_FIRMWARE_DOWNGRADE	OE.INITIALIZATION, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY
T.STORAGE_CORRUPTION	O.OPERATION, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, O.TA_AUTHENTICITY, OE.INITIALIZATION, OE.ROLLBACK, OE.TRUSTED_HARDWARE
OSP.INTEGRATION_CONFIGURATION	OE.INTEGRATION_CONFIGURATION
OSP.SECRETS	OE.SECRETS
A.PROTECTION_AFTER_DELIVERY	OE.PROTECTION_AFTER_DELIVERY
A.ROLLBACK	OE.ROLLBACK
A.TA_DEVELOPMENT	OE.TA_DEVELOPMENT
A.CONFIGURATION	OE.CONFIGURATION
A.CONNECT	OE.TRUSTED_HARDWARE, OE.TRUSTED_FIRMWARE
A.PEER.FUNC	OE.TRUSTED_HARDWARE, OE.TRUSTED_FIRMWARE
A.PEER.MGT	OE.TRUSTED_HARDWARE, OE.TRUSTED_FIRMWARE
A.CAR_PHYSICAL_PROTECTION	OE.CAR_PHYSICAL_PROTECTION
A.SECURE_INITIALIZATION	OE.INITIALIZATION
A.RPMB	OE.RPMB

Table 7 Mapping of the objectives to threats, OSPs or assumptions

The following table shows the mapping of the threats, OSPs or assumptions to objectives

Objective	**Threat, OSPs, Assumptions**

O.CA_TA_IDENTIFICATION	T.IMPERSONATION
O.KEYS_USAGE	T.ABUSE_FUNCT
O.OPERATION	T.ABUSE_FUNCT T.IMPERSONATION T.ROGUE_CODE_EXECUTION T.PERTURBATION T.STORAGE_CORRUPTION
OE.RNG	T.RNG
O.RUNTIME_CONFIDENTIALITY	T.ABUSE_FUNCT T.CLONE T.ROGUE_CODE_EXECUTION T.PERTURBATION T.RAM T.RNG T.SPY T.IMPERSONATION
O.RUNTIME_INTEGRITY	T.ABUSE_FUNCT T.CLONE T.IMPERSONATION T.ROGUE_CODE_EXECUTION T.PERTURBATION T.RAM T.RNG
O.TA_AUTHENTICITY	T.ABUSE_FUNCT T.ROGUE_CODE_EXECUTION T.PERTURBATION T.STORAGE_CORRUPTION
O.TA_ISOLATION	T.PERTURBATION T.RAM T.SPY
O.TEE_DATA_PROTECTION	T.ABUSE_FUNCT T.CLONE T.ROGUE_CODE_EXECUTION T.PERTURBATION T.STORAGE_CORRUPTION
O.TEE_ISOLATION	T.ABUSE_FUNCT T.PERTURBATION T.RAM T.SPY
O.TRUSTED_STORAGE	T.CLONE T.FLASH_DUMP T.ROGUE_CODE_EXECUTION

	T.SPY T.STORAGE_CORRUPTION
OE.INITIALIZATION	T.ABUSE_FUNCT T.CLONE T.ROGUE_CODE_EXECUTION T.PERTURBATION T.RAM T.RNG T.TEE_FIRMWARE_DOWNGRADE T.STORAGE_CORRUPTION A.SECURE_INITIALIZATION
OE.INTEGRATION_CONFIGURATION	T.ROGUE_CODE_EXECUTION T.TEE_FIRMWARE_DOWNGRADE OSP.INTEGRATION_CONFIGURATION
OE.PROTECTION_AFTER_DELIVERY	T.ROGUE_CODE_EXECUTION T.TEE_FIRMWARE_DOWNGRADE A.PROTECTION_AFTER_DELIVERY
OE.ROLLBACK	T.STORAGE_CORRUPTION A.ROLLBACK
OE.RPMB	A.RPMB T.STORAGE_CORRUPTION
OE.UNIQUE_TEE_ID	T.CLONE
OE.TA_DEVELOPMENT	T.ABUSE_FUNCT A.TA_DEVELOPMENT
OE.CAR_PHYSICAL_PROTECTION	T.PERTURBATION T.RAM T.SPY A.CAR_PHYSICAL_PROTECTION
O.TEE_ID	T.CLONE

Table 8 Mapping of the threats, OSPs or assumptions to objectives

5 Extended components definition

This Security Target defines a single extended component taken from section 6.1.3.3 of [GP TEE PP].

5.1 Extended Component AVA_TEE.2

Description:

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

The evaluator performs penetration testing on the TEE to confirm that the potential vulnerabilities cannot be exploited in the operational environment of the TEE. Penetration testing is performed by the evaluator assuming an attack potential of TEE-Low.

Definition:

AVA_TEE.2 TEE vulnerability analysis

AVA_TEE.2.1D The developer shall provide the TOE for testing.

AVA_TEE.2.1C The TOE shall be suitable for testing.

AVA_TEE.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_TEE.2.2E The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_TEE.2.3E The evaluator shall perform an independent vulnerability analysis of the TOE using the guidance documentation, functional specification, TOE design and security architecture description to identify potential vulnerabilities in the TOE.

AVA_TEE.2.4E 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 TEE-Low attack potential.

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

6 Security requirements

This chapter describes the security requirements to the TOE comprising the security functional requirements and the assurance requirements.

Additionally, following they are defined the concepts that are being used along the definition of the Security Requirements.

The subjects, objects and operations listed next, are labelled following a self-defined pattern, that is composed by <item_type>.<item_label>.<security_attribute_label> that is also used along SFRs description in section 6.1, where:

- item_type: described as “S” for subjects items, “OB” for objects items, “I” for information items and “OP” for operations items.
- item_label: a representative label of the item that is being described.
- security_attribute_label: when included, is a representative label of the security attribute being described belonging to the item referenced by <item_label>. In this case

TOE Users

Users stand for entities outside the TOE, being the following:

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

TOE Roles

- TSF: this is the role which represents the abstract entity TSF to which the actual TSF security attributes and operations can be related. Therefore, some operations performed by the TSF can be modeled as being performed by the role "TSF".
- TA_User: this role represents the TSF running on behalf of a TA, upon request from the REE (by Client Applications) or from other TAs.

TOE Subjects

Subjects stand for active entities inside the TOE, being the following:

- 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 subjects (refer to 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.
- S.ROOT_SERVER_PROCESS: the user acting on behalf of the root server.

TOE Objects

Objects stand for passive entities inside the TOE, being the following:

- 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 TOE objects:

- OB.TA_KEY (user data): (handle to a) user key (persistent or transient), with security attributes "usage", "owner" (TA identifier), "isExtractable" (True/False).
  - **Application Note**: This is an internal key that is generated by the application when key generation is requested through the GP API interface.
- OB.TEK_KEY (TSF data): the key used to decrypt the TA code, generated externally by the TA developer and is embedded (and ciphered) in the TA metadata.

TOE Informations

Information stands for data exchanged between subjects, being the following:

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

TOE Operations

Cryptographic operations on user keys performed by S.API on behalf of S.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 S.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 S.TA_INSTANCE .
- OP.ACCESS_TEK: the root server retrieves the signed TA header, manifest, encrypted ELF binary, and the corresponding certificate and signature. Then, the Root Server verifies the signature, and only if this verification succeeds, it decrypts the ELF binary.

TOE Security Functional Policies

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

- Runtime Data Information Flow Control SFP:
  - Purpose: To control the flow of runtime data from and to executable entities and memory. This policy contributes to ensure the integrity and confidentiality of runtime data
  - Subjects: S.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

- TA Keys Access Control SFP:
  - Purpose: To control the 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 the 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.

- TEK Access Control SFP:
  - Purpose: Limit the access to the decrypted TEK value only to the TSF itself and prevent the disclosure of it to any other entity of the TOE.
  - Subjects: S.ROOT_SERVER_PROCESS
  - Objects: OB.TEK_KEY
  - Security attributes: none.
  - Operations: OP.ACCESS_TEK
  - SFR instances: FDP_ITC.1/TEK, FDP_ACC.1/TEK, FDP_ACF.1/TEK

6.1 Security functional requirements

This section defines the Security functional requirements (SFRs) and the Security assurance requirements (SARs) that fulfill the TOE. The SFRs within this ST adheres to the following conventions:

- Assignments: They appear between square brackets. The word “assignment” is maintained and all together with the resolution are presented in **boldface**.
- Selections: They appear between square brackets. The word “selection” is maintained and all together with the resolution are presented in **boldface**.
- Iterations: It includes “/” and an “identifier” following requirement identifier (all in **boldface**) that allows to distinguish the iterations of the requirement. Example: **FCS_COP.1/Symmetric**.
- Refinements: Additional text appear in **boldface** with no additional square brackets.

6.1.1 In the Global Platform PP

6.1.1.1 Identification

FIA_ATD.1 User attribute definition

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users: **CA_identity, TA_identity, TA_properties, [assignment: none]**.

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.

FIA_UID.2 User identification before any action

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

Application Note:

User stands for Client Application or Trusted Application.

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.**
- **[assignment: none].**

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**
- **[assignment: If the client is a CA, then the client_identity must indicate a CA].**

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

- **No modification of client_identity is allowed after initialization.**
- **[assignment: none].**

Application Note:

TEE Internal Core 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**

- **[assignment: none].**

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

6.1.1.2 Confidentiality, Integrity and Isolation

FDP_IFC.2/Runtime Complete information flow control

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

- **Subjects: S.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.

TOE environment: Underlying Hardware and Secure Firmware must provide correct MMU operation and correct DRAM isolation configuration. (Covered in A.PEER.FUNC and A.PEER.MGT).

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 **[assignment: none]**.

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

- **Rules for information flow from and to S.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 TOE 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, 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
- The TOE is used to configure the appropriate hardware mechanism (MMU) to enforce this security functional requirement.

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: **TOE and TA objects available in runtime.**

Application Note:

This operation applies in particular upon:

- Failure detection (refer to FPT_FLS.1)
- TA instance and TA session closing.

6.1.1.3 Cryptography

FCS_COP.1/Internal Cryptographic operation

FCS_COP.1.1/Internal The TSF shall perform **[assignment: list of cryptographic operations listed in Table 9]** in accordance with a specified cryptographic algorithm **[assignment: listed in Table 9]** and cryptographic key sizes **[assignment: corresponding key sizes in Table 9]** that meet the following: **[assignment: corresponding list of standards in Table 9]**.

Operation	Cryptographic algorithm	Supported key sizes	Corresponding standards
Symmetric Cipher	AES (CBC)	256	FIPS 197 (AES) NIST SP800-38A (CBC, CTR)
Message Authenticated Code	HMAC SHA256	Limited by the block size of the hash function	RFC 4231
Digital Signature verification and decryption	RSASSA (PKCS#1 v1.5[3], PSS) RSAES (PKCS#1 v1.5[3], OAEP)	2048	PKCS#1
Digest	SHA256	Not applicable	FIPS 180-4

Table 9 FCS_COP.1/Internal details

Application Note:

TA code is verified using an RSA-2048 signature (using SHA256 digest) and decrypted using AES-256 in CBC mode. The consistency and confidentiality of Trusted Storage data is protected using a combination of AES-256 in CBC mode and HMAC SHA-256.

FCS_COP.1/GP-API Cryptographic operation

FCS_COP.1.1/GP-API The TSF shall perform **[assignment: list of cryptographic operations listed in Table 2]** in accordance with a specified cryptographic algorithm **[assignment: and their corresponding modes listed in Table 2]** and cryptographic key sizes **[assignment: corresponding key sizes in Table 2]** that meet the following: **[assignment: corresponding list of standards in Table 2]**.

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 TOE**
- **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 authorise access of subjects to objects based on the following additional rules: **[assignment: 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 TOE that is not S.API**
- **Any access to a user key attempted from S.API without valid caller (S.API.caller is undefined)**
- **[assignment: none].**

Application Note:

This requirement states access conditions to keys through the TEE Internal Core 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 Core 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 initialisation

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, [assignment: none]** to specify alternative initial values to override the default values when an object or information is created.

6.1.1.4 Initialization, Operation and Firmware Integrity

FAU_ARP.1 Security alarms

FAU_ARP.1.1 The TSF shall take **the following actions** upon detection of a potential security violation:

- **detection of consistency violation of TA data or TA code: [assignment: return an error in case of TA data, abort the execution of the TA instance in case of TA code].**
- **[assignment: none].**

FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor **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: **[assignment: MAC on TA data and keys and signature of TA code]**.

Application Note:

This SFR applies to 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.

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall:

- **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**
  - **[assignment: Return an error independent of the compromised data]**
- **[assignment: none].**

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. The TOE only protects the consistency of the TEE runtime data by logically separating the access to the TEE runtime data to authorized entities. For consistency protection against modification by means other than logical, the TOE relies on the hardware platform. 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 [IAPI], Section 2.3.3)**
- **TA code, TA data or TA keys authenticity or consistency failure**

- **[assignment: none].**

Application Note:

- Device binding failure occurs when (part of) the stored data has not been bound by the same TEE
- The secure state consists either of the halting of the TSF, rendering any access to sensitive data impossible, aborting the TA instance, or the returning of an error as described in FAU_ARP.1.

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 authorised users.**

FPT_TEE.1 Testing of external entities

FPT_TEE.1.1 The TSF shall run a suite of tests **prior execution and [assignment: none]** 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**.

6.1.1.5 TEE Identification

FAU_SAR.1 Audit review

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

6.1.1.6 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: **[assignment: none]**.

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

- **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 **[assignment: moment that the failed operation occurs].**

Application Note:

This SFR enforces atomicity of any write operation [IAPI].

The roll-back operation is transparent to the user, and not a user-initiated action.

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 initialisation

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

FMT_MSA.3.2/Trusted Storage The TSF shall allow the **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.

Application Note:

This requirement considers that user data being protected during transmission are under control of the TSF, and therefore, the TSF (and because of this, the TOE) can be extended to the physically-separated part where the data are stored.

6.1.2 SFRs additional to the PP

FCS_CKM.1/GP-API Cryptographic key generation

FCS_CKM.1.1/GP-API The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **[assignment: for the identified cryptographic algorithms listed in Table 10]** and specified **[assignment: cryptographic key sizes listed in Table 10]** that meet the following: **[assignment: corresponding list of standards in Table 10].**

Type	Cryptographic algorithm	Supported key sizes	Corresponding standards
Asymmetric Cipher	RSA	2048, 3072	NIST SP 800-133 Rev. 2 FIPS 186-4

			PKCS#1
	DSA	2048, 3072	FIPS 186-4
	ECDSA	256, 384, 521	NIST SP 800-133 Rev. 2 FIPS 186-4
Key Exchange (Shared secret derivation)	DH	2048 bits, multiple of 8 bits.	PKCS #3
	ECDH	256, 384, 521	NIST SP800-56A

Table 10 Key generation algorithms

FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **[assignment: overwriting with a fixed value]** that meets the following: **[assignment: none]**.

FDP_ITC.1/TEK Import of user data without security attributes

FDP_ITC.1.1 The TSF shall enforce the **[assignment: the confidentiality of the decrypted TEK key embedded in the TA metadata]** when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.1.2 The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.

FDP_ITC.1.3 The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: **[assignment: none]**.

Application note: The TEK key is used to decrypt the TA code. TEK key is generated by the TA developer and is embedded (and ciphered) in the TA metadata.

FDP_ACC.1/TEK Subset access control

FDP_ACC.1/TEK The TSF shall enforce the **[assignment: the TEK Access Control SFP]** on **[assignment:**

- **Subjects: S.ROOT_SERVER_PROCESS**
- **Objects: OB.TEK_KEY**
- **Operations: OP.ACCESS_TEK**

].

FDP_ACF.1/TEK Security attribute based access control

FDP_ACF.1.1/TEK The TSF shall enforce the **[assignment: TEK Access Control SFP]** to objects based on the following: **[assignment: role assigned to the TSF, and the TEK key object]**.

FDP_ACF.1.2/TEK The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **[assignment: only the TSF shall have access to the decrypted value of the TEK with the purpose of decrypting its related TA code. The value of any TEK shall not be disclosed to any other user or entity of the TOE].**

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

FDP_ACF.1.4/TEK The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[assignment: none]**.

6.2 Security assurance requirements

The development and the evaluation of the TOE shall be done in accordance to the following security assurance requirements: **EAL3 + ALC_FLR.1**.

The following table shows the assurance requirements by reference the individual components in [CC31R5P3]:

Assurance class	Assurance components
ASE: Security Target evaluation	ASE_CCL.1: Conformance claims ASE_ECD.1: Extended components definition ASE_INT.1: ST introduction ASE_TSS.1: TOE summary specification ASE_OBJ.2: Security objectives ASE_REQ.2: Derived security requirements ASE_SPD.1: Security problem definition
ALC: Life-cycle support	ALC_CMC.3: Authorisation controls ALC_CMS.3: Implementation representation CM coverage ALC_DEL.1: Delivery procedures ALC_DVS.1: Identification of security measures ALC_LCD.1: Developer defined life-cycle model ALC_FLR.1: Basic flaw remediation
ADV: Development	ADV_ARC.1: Security architecture description ADV_FSP.3: Functional specification with complete summary ADV_TDS.2: Architectural design
AGD: Guidance documents	AGD_OPE.1: Operational user guidance AGD_PRE.1: Preparative procedures
ATE: Tests	ATE_COV.2: Analysis of coverage ATE_DPT.1: Testing: basic design ATE_FUN.1: Functional testing ATE_IND.2: Independent testing - sample
AVA: Vulnerability assessment	AVA_VAN.2: Vulnerability analysis

Table 11 Security assurance requirements

6.3 Security requirements rationale

6.3.1 Rationale for the security functional requirements

The following gives an overview how the security functional requirements are combined to meet the security objectives.

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

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

- FCS_COP.1/GP-API specifies the allowed operations.
- FCS_CKM.1/GP-API specify key generation for these operations.
- FCS_CKM.4 specifies key destruction for these operations.
- 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.

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

- FAU_SAR.1 provides TA access to the unique identifier provided by the hardware.

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

- FAU_ARP.1 states the TOE responses to potential security violations.
- FDP_SDI.2 enforces the monitoring of consistency and authenticity of TA, and it states the behavior in case of failure.
- FIA_ATD.1, FIA_UID.2 and FIA_USB.1 ensure that actions are performed by identified users.
- FMT_SMR.1 states the two operational roles enforced by the TOE.
- 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 TOE

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.

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

- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime ensure read access to authorized entities only.

- FDP_RIP.1/Runtime states resource clean up policy.

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

- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state TOE and TA runtime data policy, which grants write access to authorized entities only.

- FDP_SDI.2 monitors the authenticity and consistency of TA code and the TA data and keys and states the response upon failure.

O.TA_AUTHENTICITY The following requirements contribute to fulfill 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 execution.

- FCS_COP.1/Internal states the cryptography used to verify the authenticity of TA code. FDP_ITC.1/TEK, FDP_ACC.1/TEK, FDP_ACF.1/TEK contribute in defining and establishing a clear boundary on the imported key (TEK) for decrypting each TA code in the TOE, which shall be only accessible by the TSF and no other entity of the TOE.

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

- FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage and FMT_SMF.1 state the policy for controlling access to TA storage.

- FCS_COP.1/Internal state the cryptographic algorithms used for Trusted Storage to ensure confidentiality and authenticity 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, in particular in case of panic states.

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

- FCS_COP.1/Internal states the cryptography used to protect consistency and confidentiality of the TOE data in external memory.

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

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

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

- FCS_COP.1/Internal states the cryptography used to protect integrity and confidentiality of the TA data in external memory, if applicable
- FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage and FMT_SMF.1/Trusted 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.
- FDP_ITT.1/Trusted Storage ensure protection against disclosure of TOE and TA data that is transferred between resources
- FPT_FLS.1 maintains a secure state.

The following table maps the security functional requirements to security objectives for the TOE:

Objective	**SFRs**
O.CA_TA_IDENTIFICATION	FIA_ATD.1 FIA_UID.2 FIA_USB.1
O.KEYS_USAGE	FCS_COP.1/GP-API FDP_ACC.1/TA_keys FDP_ACF.1/TA_keys FMT_MSA.1/TA_keys FMT_MSA.3/TA_keys FMT_SMR.1 FMT_SMF.1 FCS_CKM.1/GP-API FCS_CKM.4
O.TEE_ID	FAU_SAR.1
O.OPERATION	FAU_ARP.1 FDP_SDI.2 FIA_ATD.1 FIA_UID.2 FIA_USB.1 FMT_SMR.1 FPT_FLS.1 FDP_ACC.1/TA_keys FDP_ACC.1/Trusted Storage FDP_ACF.1/TA_keys FDP_ACF.1/Trusted Storage

	FMT_MSA.1/TA_keys FMT_MSA.1/Trusted Storage FMT_MSA.3/TA_keys FMT_MSA.3/Trusted Storage FMT_SMF.1 FDP_IFC.2/Runtime FDP_IFF.1/Runtime
O.RUNTIME_CONFIDENTIALITY	FDP_IFC.2/Runtime FDP_IFF.1/Runtime FDP_RIP.1/Runtime
O.RUNTIME_INTEGRITY	FDP_IFC.2/Runtime FDP_IFF.1/Runtime FDP_SDI.2
O.TA_AUTHENTICITY	FDP_SDI.2 FPT_TEE.1 FCS_COP.1/Internal FDP_ITC.1/TEK FDP_ACC.1/TEK FDP_ACF.1/TEK
O.TA_ISOLATION	FDP_ACC.1/Trusted Storage FDP_ACF.1/Trusted Storage FMT_MSA.1/Trusted Storage FMT_MSA.3/Trusted Storage FMT_SMF.1 FCS_COP.1/Internal FDP_IFC.2/Runtime FDP_IFF.1/Runtime FPT_FLS.1
O.TEE_DATA_PROTECTION	FCS_COP.1/Internal
O.TEE_ISOLATION	FDP_IFC.2/Runtime FDP_IFF.1/Runtime
O.TRUSTED_STORAGE	FCS_COP.1/Internal 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 FDP_ITT.1/Trusted Storage FMT_SMF.1 FDP_SDI.2FPT_FLS.1

Table 12 Mapping of SFRs to TOE objectives

The following table maps the security objectives for the TOE to security functional requirements:

SFR	Objective
FAU_ARP.1	O.OPERATION
FAU_SAR.1	O.TEE_ID
FCS_COP.1/GP-API	O.KEYS_USAGE
FCS_COP.1/Internal	O.TA_AUTHENTICITY O.TA_ISOLATION O.TEE_DATA_PROTECTION O.TRUSTED_STORAGE
FCS_CKM.1/GP-API	O.KEYS_USAGE
FCS_CKM.4	O.KEYS_USAGE
FDP_ACC.1/TA_keys	O.KEYS_USAGE O.OPERATION
FDP_ACC.1/Trusted Storage	O.OPERATION O.TA_ISOLATION O.TRUSTED_STORAGE
FDP_ACC.1/TEK	O.TA_AUTHENTICITY
FDP_ACF.1/TA_keys	O.KEYS_USAGE O.OPERATION
FDP_ACF.1/Trusted Storage	O.OPERATION O.TA_ISOLATION O.TRUSTED_STORAGE
FDP_ACF.1/TEK	O.TA_AUTHENTICITY
FDP_IFC.2/Runtime	O.OPERATION O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY O.TA_ISOLATION O.TEE_ISOLATION
FDP_IFF.1/Runtime	O.OPERATION O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY O.TA_ISOLATION O.TEE_ISOLATION
FDP_ITC.1/TEK	O.TA_AUTHENTICITY
FDP_ITT.1/Trusted Storage	O.TRUSTED_STORAGE

FDP_RIP.1/Runtime	O.RUNTIME_CONFIDENTIALITY
FDP_ROL.1/Trusted Storage	O.TRUSTED_STORAGE
FDP_SDI.2	O.OPERATION O.RUNTIME_INTEGRITY O.TA_AUTHENTICITY O.TRUSTED_STORAGE
FIA_ATD.1	O.CA_TA_IDENTIFICATION O.OPERATION
FIA_UID.2	O.OPERATION O.CA_TA_IDENTIFICATION
FIA_USB.1	O.OPERATION O.CA_TA_IDENTIFICATION
FMT_MSA.1/TA_keys	O.KEYS_USAGE O.OPERATION
FMT_MSA.1/Trusted Storage	O.OPERATION O.TA_ISOLATION O.TRUSTED_STORAGE
FMT_MSA.3/TA_keys	O.KEYS_USAGE O.OPERATION
FMT_MSA.3/Trusted Storage	O.OPERATION O.TA_ISOLATION O.TRUSTED_STORAGE
FMT_SMF.1	O.KEYS_USAGE O.OPERATION O.TA_ISOLATION O.TRUSTED_STORAGE
FMT_SMR.1	O.KEYS_USAGE O.OPERATION
FPT_FLS.1	O.OPERATION O.TA_ISOLATION O.TRUSTED_STORAGE
FPT_TEE.1	O.TA_AUTHENTICITY

Table 13 Mapping of TOE objectives to SFRs

6.3.2 Dependencies of security functional requirements

The Table below lists the security functional requirements defined in this Security Target, their dependencies and whether they are satisfied by other security requirements defined in this Security Target.

SFR	Dependencies	Fulfilled

FIA_ATD.1	None	Not applicable
FIA_UID.2	None	Not applicable
FIA_USB.1	FIA_ATD.1	FIA_ATD.1
FMT_SMR.1	FIA_UID.1	FIA_UID.2
FDP_IFC.2/Runtime	FDP_IFF.1	FDP_IFF.1/Runtime
FDP_IFF.1/Runtime	FDP_IFC.1 and FMT_MSA.3	FDP_IFC.2/Runtime Justified below (1)
FDP_RIP.1/Runtime	None	Not applicable
FCS_COP.1/GP-API	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and FCS_CKM.4	FCS_CKM.1/GP-API FCS_CKM.4
FCS_COP.1/Internal	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and FCS_CKM.4	FDP_ITC.1/TEK FCS_CKM.4 justified below (2)
FCS_CKM.1/GP-API	(FCS_CKM.2 or FCS_COP.1) and FCS_CKM.4	FCS_COP.1/GP-API FCS_CKM.4
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	FCS_CKM.1/GP-API
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 and FMT_SMR.1	FDP_ACC.1/TA_keys FMT_SMF.1 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	Justified below (3)
FDP_SDI.2	None	Not applicable
FPT_FLS.1	None	Not applicable

FMT_SMF.1	None	Not applicable
FPT_TEE.1	None	Not applicable
FAU_SAR.1	FAU_GEN.1	Justified below (4)
FDP_ACC.1/Trusted Storage	FDP_ACF.1	FDP_ACF.1/Trusted Storage
FDP_ACF.1/Trusted Storage	FDP_ACC.1 and FMT_MSA.3	FDP_ACC.1/Trusted Storage FMT_MSA.3/Trusted Storage
FDP_ROL.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted Storage
FMT_MSA.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1) and FMT_SMF.1 and FMT_SMR.1	FDP_ACC.1/Trusted Storage FMT_SMF.1 FMT_SMR.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_ITC.1/TEK	(FDP_ACC.1 or FDP_IFC.1) FMT_MSA.3	FDP_ACC.1/TEK FMT_MSA.3 unfulfilled. Justified below (5).
FDP_ACC.1/TEK	FDP_ACF.1	FDP_ACF.1/TEK
FDP_ACF.1/TEK	FDP_ACC.1 FMT_MSA.3	FDP_ACC.1/TEK FMT_MSA.3 unfulfilled. Justified below (6).

Table 14 SFR dependency analysis

As in the Protection Profile, the following dependencies are discarded:

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

(2) The dependency FCS_CKM.4 of FCS_COP.1/Internal is discarded. The TEE storage root of trust used for cryptographic operations in FCS_COP.1/Internal is not required to be changed or destroyed during the end-usage phase. Note that the cryptographic operations offered via the GP API do have a dependency on FCS_CKM.4, which is satisfied by FCS_CKM.4.

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

(4) The dependency FAU_GEN.1 of FAU_SAR.1 is discarded. This dependency is discarded as the only audit

record considered is the TEE identifier and this identifier is set before TOE delivery and non-modifiable afterwards.

Additionally, the following dependencies are not met:

(5) The dependency FMT_MSA.3 of FDP_ITC.1/TEK is not met. There is no management of security attributes by authorized users for the related information flow SFP. The imported key is under exclusive control of the TSF and none of its security attributes can be managed by entities other than the TSF itself.

(6) The dependency FMT_MSA.3 of FDP_ACF.1/TEK is not met. There is no management of security attributes by authorized users for the related information flow SFP. The imported key is under exclusive control of the TSF and none of its security attributes can be managed by entities other than the TSF itself.

6.3.3 Rationale for the security assurance requirements

The selected EAL permits a developer to gain sufficient assurance from positive security engineering based on good TEE commercial development practices that are compatible with industry constraints, in particular 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 standard EAL3.

Finally, the inclusion of ALC_FLR.1 aligns with the principle of continuous improvement, a fundamental aspect of security management by recognizing that security is an ongoing process which requires the identification and resolution of future flaws by reducing the window of opportunity for potential attackers to exploit vulnerabilities.

6.3.4 Dependencies of security assurance requirements

Dependency analysis is fulfilled as it has been chosen a predefined CC package described in section 8.7 of [CC31R5P3] which dependencies are directly solved.

7 TOE Summary specification

7.1 Isolation of TEE and REE, and of TAs

By implementing isolated execution environments that isolate the TEE and REE, and by isolating the TAs from the other TEE services, the TOE implements the SFRs related to 6.1.1.2 Confidentiality, Integrity and Isolation, (FDP_IFC.2/Runtime, FDP_IFF.1/Runtime and FDP_RIP.1/Runtime).

The TOE itself is an operating system working on a secure side as defined in [SA]. The TOE is composed of a kernel (microkernel architecture) and services and other resources. To achieve isolation between TEE and REE sides the ARM TrustZone technology is utilized. This technology allows to share hardware resources at the same time providing full isolation between systems running on TEE and REE sides. To provide isolation within the TEE side, the kernel implements a concept of process. Different processes work in isolated address spaces. The isolation of processes is achieved mainly by hardware means such as MMU (memory management unit), exception levels and modes of operation. TAs and services are running as separate processes. Any interaction between subjects (TA instances, RAM, resources, communication manager) is strictly controlled by the kernel and one of the services called Root Server. Root Server is a service responsible for management of TA instances. Communication between REE and TEE sides is achieved by SMC (secure monitor call) invocations and memory areas shared between REE and TEE sides. The communication is mediated by the kernel itself and routed to Root Server and destination TAs.

The memory allocated on behalf of processes (TAs, services) is also managed by the kernel. When a TA requests a new memory area, before passing it to the TA, the kernel overwrites this fragment of memory with a constant pattern. This way all residual information is erased (FDP_RIP.1/Runtime).

7.2 Protected communication interface between CAs and TAs

The communication interface between CAs and TAs identifies every entity that is part of the communication, thus ensuring that the identity is bound to the communicating entity.

The TSF in this case will identify and authenticate CA according to white CA list of TA, and identify and authenticate TA by its signature, any further actions performing by CA/TA user must be prevent before successful authentication. Both identifiers of CA and TA cannot be modified during their life cycle.

However, the REE side is responsible for managing the CA identity, and the TEE side does not necessarily trust the information provided by the REE. This implements the SFRs related to 6.1.1.1 Identification, (FIA_ATD.1, FIA_UID.2, FIA_USB.1 and FMT_SMR.1).

7.3 Trusted storage of TA and TOE data and keys

The Trusted Storage server is a service running inside the TOE, which implements the SFRs related to 6.1.1.6 Trusted Storage; (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 FDP_ITT.1/Trusted Storage) together with the roles described in FMT_SMR.1 for the particular case of the functions related to the security attributes.

The trusted Storage Server guarantees authenticity, integrity, and confidentiality by implementing appropriate cryptographic functionality (FCS_COP.1/Internal) to encrypt and authenticate data in the Trusted Storage with inputs derived from the TA identity and TEE identity, thus additionally preventing other TAs to access this data and binding the data to the TEE. This binding is achieved by the encryption key used to encrypt the trusted storage data, which is derived from a master key using the unique TEE identifier. Additionally, each TA code is encrypted with key to ensure the confidentiality and authenticity by the use of the TEK key (generated by each TA developer). FDP_ITC.1/TEK, FDP_ACC.1/TEK and FDP_ACF.1/TEK provide support in keeping that key under the sole control and access of the TSF only.

The unique TEE identifier can be retrieved by the TAs, which implements the SFRs related to 6.1.1.5 TEE Identification, (FAU_SAR.1). The TA identity is used to enforce that data belonging to a specific TA cannot be accessed by other TAs, which implements the SFRs related to 6.1.1.5 TEE Identification, (FIA_ATD.1, FIA_UID.2 and FIA_USB.1).

The storage basic rollback is achieved by using rename operation when creating and modifying an object. This implements the SFR related to 6.1.1.6 Trusted Storage, (FDP_ROL.1/Trusted Storage). The unique identifier (FAU_SAR.1) is provided by the hardware platform which is stored in OTP memory and is accessible via API exposed to TAs.

7.4 Cryptographic API

By implementing the cryptographic functionality described in the [IAPI] in addition to the cryptographic functionality used to secure the other internal TOE functions, the TOE implements the SFRs related to 6.1.1.3 Cryptography; (FCS_COP.1/GP-API, 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) and related to 6.1.2 SFRs additional to the PP, (FCS_CKM.1/GP-API and FCS_CKM.4).

7.5 TA instantiation

TA instantiation is performed by a Root Server running as a service inside the TEE. This Root server implements part of the SFRs described in 6.1.1.4 Initialization, Operation and Firmware Integrity, (FPT_FLS.1, FAU_ARP.1, FDP_SDI.2 and FPT_TEE.1), related to verifying the integrity of the TA code and handling potential verification failures. This is achieved by a digital signature on the TA code (which implements a part of FCS_COP.1/Internal) coupled with unique identifiers for each TA.

Additionally, TA code is encrypted. Apart from that the authenticity and consistency of TOE code, and TOE runtime data is achieved by the environment (OE.INITIALIZATION): the secure bootloader and the hardware platform.

7.6 Correct execution of TOE

The correct execution of the TOE is achieved by implementing secure initialization procedures and various verifications at runtime. The initialization process guarantees the TSF integrity from the power-off state into an initial secure state. The initialization process includes the chipset and low level software layer initialization (A.SECURE_INITILIZATION, OE.INITIALIZATION) and will transfer control over to the TOEs secure initialization process whereby the TOEs self-protection mechanisms are enabled. The TOE reaches a secure state after a successful completion of the initialization sequence in proper order.

During runtime, the TOE performs various checks such as: detection of invalid CA requests (bad-formed requests), detection of failure of cryptographic operations, detection of panic states (as defined in [IAPI], Section 2.3.3).

Before running of a TA, the TOE performs verification of authenticity and consistency of TA data, code and keys as enforced by FPT_TEE.1. Additionally, FCS_COP.1/Internal states the cryptography used to verify the authenticity of TA code. The TOE prevents from running unauthenticated and inconsistent TAs. Upon verification of such a TA, an appropriate error code is returned to a calling CA.

Detection of any of the above-mentioned failures preserves a secure state. This functionality implements the remaining parts of the SFRs described in 6.1.1.4 Initialization, Operation and Firmware Integrity, (FPT_FLS.1, FAU_ARP.1 and FDP_SDI.2).

8 Appendices

8.1 Appendix A – TRNG (True Random Number Generator) System

8.1.1 Overview

Figure 4 illustrates block diagram of TRNG:

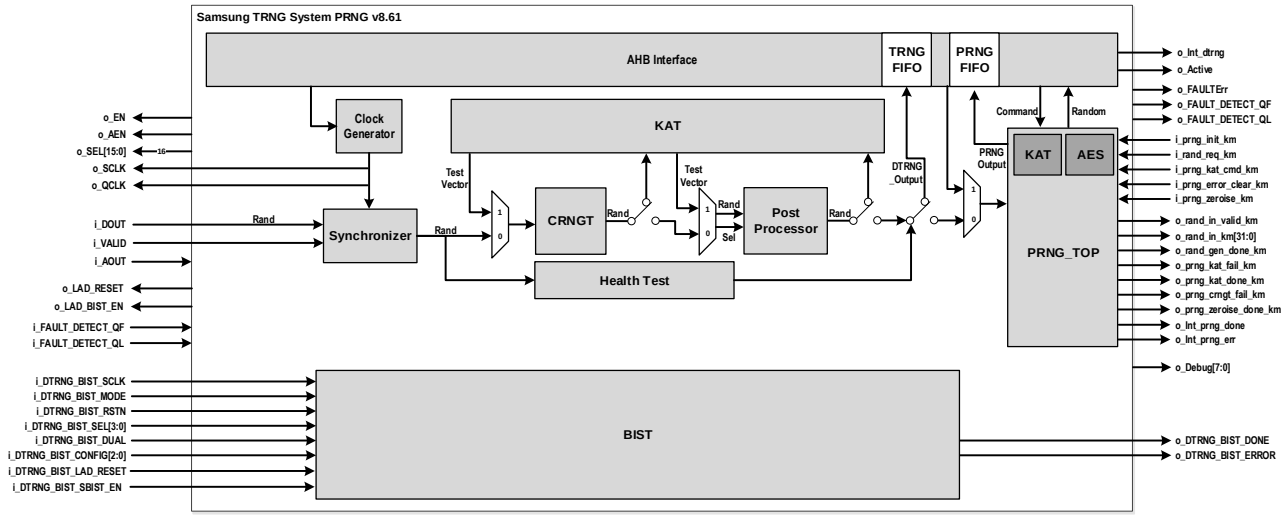

Figure 4 Block Diagram of TRNG

True Random Number Generator (TRNG) generates random numbers from an amplified and sampled thermal noise of a chip. The generated random numbers are non-deterministic and unpredictable bit stream. To satisfy different requirements of the customers, TRNG module includes several TRNG circuits that are designed with different characteristics (performance, low power consumption and so on).

Using the seed either from the user seed or from TRNG seed, the PRNG IP generates random number. You can select which seed to be used for the PRNG outputs. The PRNG IP is based on AES-based algorithm based on NIST 800-90A specification for generating the random numbers. The PRNG IP provides a key manager IP with random numbers. The Key manager IP uses random numbers that the PRNG IP provides for secret key and salt values.

Figure 5 illustrates basic principle of TRNG operation:

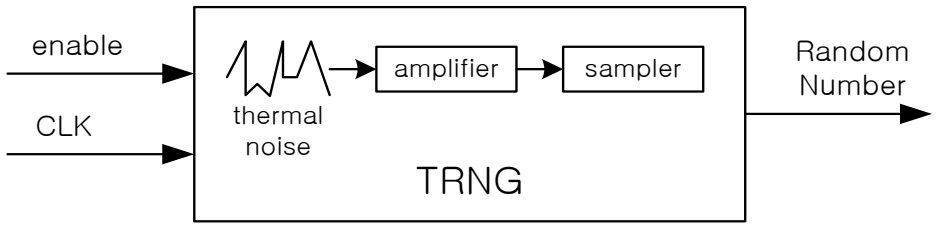

Figure 5 Principle of TRNG

8.1.2 Functional Description

The TRNG generates random bits from thermal noise. The output of TRNG might be independent and biased because of possible imperfection of TRNG or instability of environment (fabrication process variation, oversampling, time degradation, and so on), an embedded post-processor can remove this statistical weakness. Health Test can be dedicated to verify the quality of produced random bits during the generation. TRNG system has embedded several post-processor functions, including bypass mode.

Also TRNG has PRNG Function. PRNG includes DRBG which is compatible with NIST SP 800-90A. PRNG can generate 256 bits pseudo random.

Figure 6 illustrates the internal blocks of a TRNG system:

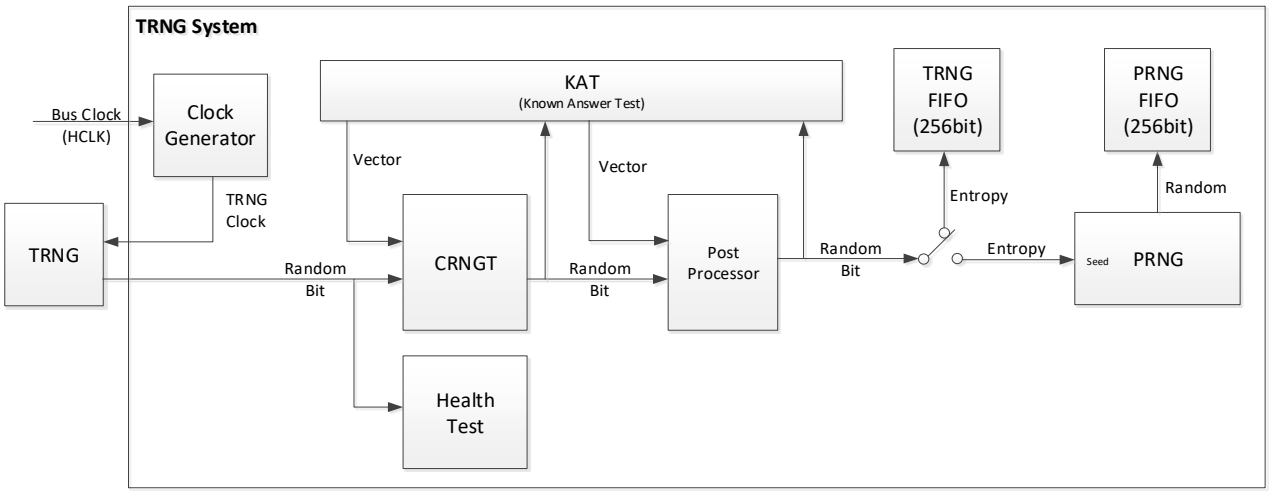

Figure 6 Block Diagram of a TRNG System

Table 15 describes the internal blocks of a TRNG system:

Block	Input	Output	Control	Function
Clock Generator	Bus Clock (ACLK)	TRNG Clock	Divisor Pulse Width	This block divides input clock to generate slower TRNG clock. If the TRNG clock is too fast for the particular TRNG circuit, the quality of random number may be poor. For most TRNG circuits, the clock frequency below of 500KHz would be safe. The divisor should be even number. Also user can control duty cycle ratio of clock between high and low. Also user can set clock ratio between high and low through Pulse_Width value.

				- Output_Freq = Input_Freq/Divisor - Pulse Ratio -> High : Low = Pulse_Width : ((# of Input Cycle in 1 Output Cycle) - Pulse Width)
TRNG	TRNG Clock	Random Bit	TRNG Enable, TRNG Selection	TRNG block consists of several TRNG circuits. Among those TRNG circuits, select the appropriate circuit. (Refer to TRNG hard macro user guide for more information). The output throughput (bits per second) is similar to the input clock frequency.
KAT	CRNGT output PP output	Golden vector Test Done Test result	KAT Enable	KAT block checks the CRNGT block and Post-Processor block operation. This block delivers golden vector to CRNGT and PP. If CRNGT or PP generates an unexpected value, KAT generates a KAT_CRNGTErr or KAT_PPErr. If KAT fails, an output data must be blocked. After reset, KAT automatically runs and takes 292 cycles until complete.
CRNGT	Random bit KAT Golden Vector	Random Bit	CRNGT Enable	CRNGT block check continuous repetition of 16 bits pattern. If same 16 bits was generated from TRNG, same pattern 16 bits is thrown out. Whenever you enable CRNGT, first 16 bits is used only for comparison and not used as output. This test is compatible with FIPS 140-2.
Health Test	Random Bit	Test result	HT Enable	This test is required in NIST SP 800-90B spec. This block consists of Repetition Count Test and Adaptive Proportion Test. Whenever change RNGSEL or CLKDIV or perform a reset, TRNG should successfully pass this test at least once. If not, output data must be blocked. To complete this test, it needs a 1K Random bit.
Post Processor	Random Bit	Statistical improved Random Bit	Post Processor Enable, Post Processor Selection	This block tries to transform the biased input random bits to unbiased output. There are four types of post processors in this block. Linear Feedback Shift Register (LFSR) Von Neumann XOR Advanced Von Neumann Disable this block or select the 0th post processor to bypass this function.
TRNG FIFO	Random Bits (in 1bit)	Random Bits (in 32bit)	FIFO Pointer	The CPU can read the random number from TRNG through this FIFO. This block has a pointer into the FIFO where the new random bit should be stored. When users set this pointer to number (1 to 256), the pointer decreases automatically and random bits are stored in FIFO. When the pointer reaches the value of zero, it stops decreasing and the FIFO also stops storing new random bit.

PRNG	True Random Bit	Pseudo Random Bit	Refer to PRNG Section	The TRNG can be used for seeding the PRNG. User can force the PRNG to be seeded and the random bit automatically flows from TRNG to PRNG automatically.

Table 15 Internal Blocks of TRNG System

There are several threshold values that can be set by SFR configuration. If it is needed to change any of the default values, please contact the design team. Moreover, it is recommended to keep the default values of Online Test if the compliance of AIS.31 is desired.

8.1.2.1 Health Test Cut-off Value

8.1.2.1.1 Repetition Count Test

Repetition count test is designed using 1bit sample size. Cut-off value can control error detection level. Assume that false positive error rate is 2^{-40}, Cut-off value and min-entropy are following below equation.

$$C = 1 + \frac{40}{H}$$ (C: Cut-off value, H: Min-Entropy per 1bit)

Below is cut-off value example table. Cut-off value should be bigger than 41.

Cut-off Dec (C)	Cut-off Hex.	Min-Entropy per bit
41	9'h029	1.0
45	9'h02D	0.9
51	9'h033	0.8
58	9'h03A	0.7
67	9'h043	0.6
81	9'h051	0.5 (Default)
101	9'h065	0.4
134	9'h086	0.3
201	9'h0C9	0.2
401	9'h191	0.1

Table 16 Repetition Count Test Cut-off Value

8.1.2.1.2 Adaptive Proportion Test

Adaptive proportion test is designed using 1bit sample size. Cut-off value can control error detection level.

Cutoff value can be computed as:

$$\sum_{i=0}^{C}\binom{1024}{i}P^{i}(1-P)^{1024-i} \geq 1-2^{-40}, P = 2^{-H}$$

Cut-off (C) Dec	Cut-off Hex.	Min-Entropy per bit
625	12'h271	1.0
661	12'h295	0.9
699	12'h2BB	0.8
739	12'h2E3	0.7
780	12'h30C	0.6
824	12'h338	0.5 (Default)
869	12'h365	0.4
915	12'h393	0.3
961	12'h3C1	0.2
1005	12'h3ED	0.1

Table 17 Adaptive Proportion Test Cut-off Value

8.1.2.2 PRNG

PRNG supports NIST SP 800-90A compliant DRBG algorithm. Among several DRBG algorithms listed in the NIST SP 800-90A, PRNG supports AES-256 based CTR_DRBG function. Entropy and nonce of PRNG algorithm could be configured either SFR register or random numbers which are generated by TRNG directly without software access. Output size of PRNG is 256 bits.

CTR_DRBG function consists of three commands. Instantiate command receives entropy and nonce inputs, and updates the key and internal states of V value. Reseed command receives entropy values and updates the key and internal states of V values. Generate command generates the 256 bits of output data and update the key and V values.

Using the following configurations of PRNG, the size of the parameters used for CTR_DRBG in this PRNG is as follows.

- Algorithm: CTR DRBG (AES- 256)
- Entropy Size: 512 bits

- Nonce Size: 256 bits
- Seedlen: 384 bits Key & V size are 384 bits
- Security strength: 256 bits

Note: For detailed functional specification, refer the NIST SP 800-90A document.

8.1.2.3 PRNG Commands

The following sections describe the functions of CTR_DRBG specified in NIST SP 800-90A. Using the parameters listed above, the CTR_DRBG functions are elaborated as below

- Instantiate
- Reseed
- Generate
- Uninstatiate

8.1.2.3.1 Instantiate Command

Using the following configurations of PRNG, the sizes of the parameters for the instantiate command and the sizes of entropy and nonce for the instantiate command to be updated are as follows:

- Entropy Size: 512 bits
- Nonce Size: 256 bits

Before the instantiate command is issued, user should set the entropy and nonce values to be used during the instantiate command operation. There are two ways to set the entropy and nonce values: set the SFR and use the TRNG. Use the DTRNG data to fill the entropy and nonce values. For additional information on the sequence to use DTRNG for entropy and nonce values, refer to the Programming Guide.

8.1.2.3.2 Reseed Command

Reseed command updates the current internal state values with updated entropy values. Configure SFR or use DTRNG to set the new entropy. For additional information on the sequence to set the new entropy values for reseed function, refer the Programming Guide. The size of entropy values to be updated is as follows.

- Entropy Size: 512 bits

8.1.2.3.3 Generate Command

Use the generate command to generate pseudo random numbers. Using current internal state values which were updated with instantiation and reseed function, 256 bits of pseudo random numbers are generated. After validating if the generates command is complete, you can read the 256 bits of pseudo random data.

8.1.2.3.4 Uninstantiate Command

By performing "uninstantiate" command, key, internal state values such as V and reseed counter become "zero".

8.1.2.4 Reseed Counter

After the generate command is complete, the 32 bits hardware reseed counter is located and its counter value is incremented. If the current reseed counter value is greater than the configured reseed counter max value, interrupt is generated and the generate command is not executed and therefore, output is not generated. If the reseed counter value is not greater than the reseed interval value, the output is generated. User must issue reseed or instantiate command to make reseed counter value as "0".

8.1.2.5 Interrupts

PRNG provides the following different interrupts:

- KAT done interrupt – This interrupt occurs when the KAT command is successfully complete.
- Generate done interrupt – This interrupt occurs when the generate command is successfully complete.
- Reseed done interrupt – This interrupt occurs when the reseed command is successfully complete.
- Instantiate done interrupt – This interrupt occurs when the instantiate command is successfully complete.
- Uninstantiate done interrupt – This interrupt occurs when the uninstantiate command is successfully complete.
- Reseed counter error interrupt – When generation command is issued and current reseed counter value is greater than reseed counter interval value which was set using SFR, this interrupt occurs.
- KAT error interrupt – This interrupt occurs when the KAT command is issued.
- Instantiate error interrupt – This interrupt occurs when the instantiate command is issued but entropy and nonce values are not set completely either by SFR or DTRNG.
- Generate error interrupt – This interrupt occurs when the generate command is issued but instantiate command was not performed previously to update internal state values.
- Reseed error interrupt – This interrupt occurs when the reseed command is issued but instantiate command was not performed previously to update internal state values.

8.1.2.6 PRNG KAT Control

PRNG provides functions regarding KAT (Known Answer Test). Through the KAT command in SFR, user can run KAT automatically. After KAT, user can find the result pass or fail through SFR or interrupt.