Samsung TrustWare 3.0 (v3.0.5)

Security Target

Version: 1.0

Issue date: September 5, 2019

SAMSUNG

Version history

version	Description	Date
0.1	First draft version to support the workshop	22/02/2019
0.2	Fill out SFRs and pick objectives at workshop	26/02/2019
0.3	First draft version after workshop	01/03/2019
0.4	Added missing coverage tables and TSS	05/03/2019
0.5	Process review comments	05/03/2019
0.6	Fill out usage of TOE Updated supported cryptographic algorithms Updated References version	07/03/2019
0.7	Finalised last claims	12/03/2019
0.8	Processed review comments	15/03/2019
0.9	Final draft to be used for evaluation	15/03/2019
0.10	Added abbreviations and updated TOE reference	15/03/2019
0.11	Processed first round evaluation comments	18/04/2019
0.12	Minor textual changes and fixes, adapted Figure 1	28/05/2019
0.13	Updated References version	25/06/2019
0.14	Described SFRs in the Chap.7 in more details and updated TOE and document versions	03/07/2019
0.15	Updated References version	09/07/2019
0.16	Described the non-TOE components in more details	17/07/2019
0.17	Updated References version	24/07/2019
0.18	Added side-channel and fault injection statement and updated References version	31/07/2019
0.19	Clarification in Section 7.6	01/08/2019
0.20	Updated References version	22/08/2019
0.21	Updated References version	23/08/2019
0.22	Added security guidelines and updated References version	28/08/2019
0.23	Fixed trusted storage SFR mapping in Section 7.3	02/09/2019
1.0	Fixed typos in TOE Overview and updated logos and styles	05/09/2019

Contents

1 ST INTRODUCTION

1.1 Security Target reference

The Security Target is entitled “Samsung TrustWare 3.0 (v3.0.5) Security Target”.

For the document version and date, please see the Version history table.

1.2 TOE Reference

The TOE is identified as: TrustWare 3.0 (version 3.0.5)

1.3 TOE Overview

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

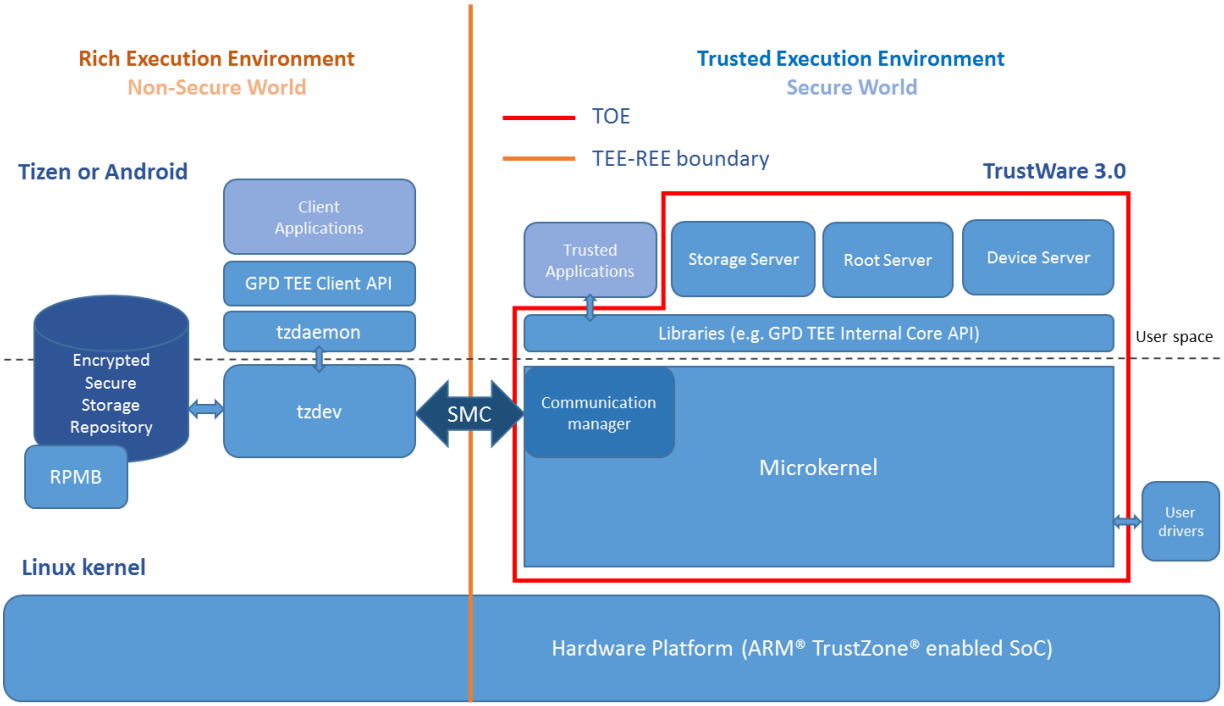

FIGURE 1 TOE OVERVIEW

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 hosts 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)

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

- Samsung DTV Platform series SoC, such as sdp1803(Muse-M), An ARM-based SoC including TrustZone functionality
- RAM, which stores TrustWare's and REE's runtime data
- ROM, which keeps secure boot
- FLASH, which keeps TrustWare and REE components (e.g. Tizen, Android)
- Secure OTP memory, which contains sensitive data required by the secure bootloader
- Necessary peripherals (e.g. display, etc.)
- Secure bootloader enabling secure boot (Samsung's Onboot bootloader, version >= 6).
- Implementation of the REE side (e.g. Tizen, Android)
- Some device-specific drivers for the TEE side such as:
    - psci_drv – used for power management of CPUs
    - se – the driver providing access to the unique device identifier, chip identifier, and the seed for the random number generator
    - tzasc – TrustZone Address Space Controller is used for protection of address regions
    - tzpc – TrustZone Protection Controller is used for assigning memory regions to secure and non-secure sides

The TOE is intended to be used in the following applications:

- DRM protection – services using a video and audio streaming or playing (e.g. streaming online video content, playing online audio content, playing offline encrypted files). Content of streaming or playing is encrypted and needs a trusted system to handle decryption, which the TOE provides.
- Online payments and online banking – payments which use a secure method to enter and store data. The payment process, which consists of the input of confidential data (like credit card numbers) by a virtual keyboard, should be performed by a trusted system. In addition, storing confidential data should be performed in a secure way
- Protection of interactive content – streaming games from producer servers should be performed by trusted system to securely copy the interactive content
- Protection of biometric data (e.g. when biometric data of facial recognition should be handled by a trusted operating system)
- Web-based services which use confidential user data like passwords and profile information should be handled by a trusted system

1.4 TOE description

1.4.1 TOE physical scope

The TOE consists of the set of software files (comprising TrustWare 3.0 (version 3.0.5)) and guidance documents that are delivered in electronic format through a file transfer. The following table lists the guidance documents and their format.

TABLE 1 GUIDANCE DOCUMENTS

Title	Version	Format
TrustWare 3.0 Certified Product Guidance	1.10	DOCX
TrustWare 3.0 Internal Driver API	1.4	PDF
TrustWare 3.0 List of Secure Monitor Calls	1.4	PDF
TrustWare 3.0 Operational User Guidance	1.7	PDF
TrustWare 3.0 Preoperative Procedures	1.7	PDF
TrustWare 3.0 TA SDK User Manual	1.15	PDF
TrustWare 3.0 TEE API Extensions	1.4	PDF
TrustWare 3.0 User-Space Drivers	1.3	DOCX
TrustWare 3.0 Driver Porting Guide	1.3	DOCX
TrustWare 3.0 Compilation Guide	1.4	DOCX
TrustWare 3.0 Installation Guide	1.3	DOCX

1.4.2 TOE logical scope

The logical security features of the TOE can be divided in the following categories following [GP TEE PP]

- Isolation of the TEE services, the TEE resources involved and all the Trusted Applications from the REE
- Isolation between Trusted Applications and isolation of the TEE from Trusted Applications
- Protected communication interface between CAs and TAs within the TEE, including communication endpoints in the TEE
- Trusted storage of TA and TEE data and keys, ensuring consistency, confidentiality, atomicity and binding to the TEE
- Random Number Generator
- Cryptographic API including:
  - Generation and derivation of keys and key pairs
  - Support for cryptographic algorithms as described in Table 2
- 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.

TABLE 2 SUPPORTED CRYPTOGRAPHIC ALGORITHMS

Cryptographic Operation	Cryptographic Algorithm	Supported key sizes	Corresponding Standards
Symmetric Cipher	AES (ECB, CBC, CTR, XTS, CCM, GCM)	128, 192, 256[1]	FIPS 197 (AES) NIST SP800-38A (ECB, CBC, CTR) IEEE Std 1619-2007 (XTS) RFC 3610 (CCM) NIST 800-38D (GCM)
	DES, TDES (ECB, CBC)	56, 112, 168	FIPS 46 (DES, 3DES) FIPS 81 (ECB, CBC)
Digest	MD5, SHA1, SHA224, SHA256, SHA384, SHA512	Not applicable	RFC 1321 (MD5) FIPS 180-4 (SHA1 SHA224 SHA256 SHA384 SHA512)
MAC	AES (CMAC, CBC MAC)	128, 192, 256	NIST SP800-38B (CMAC) ISO9797 (CBC MAC)
	DES, TDES (PKCS5, CBC MAC)	56, 112, 168	ISO9797 RFC1423
	HMAC (MD5, SHA1, SHA224, SHA256, SHA384, SHA512)	Limited by the block size of the hash function	RFC 2202 (MD5, SHA1) RFC 4231 (SHA224 SHA256 SHA384 SHA512)
Asymmetric Cipher	RSAES (without padding, PKCS#1 v1.5, OAEP)	256, 512, 768, 1024, 1536, 2048, 3072	PKCS#1
Digital Signature	RSASSA (without padding, PKCS#1 v1.5,	256, 512, 768, 1024, 1536, 2048, 3072	PKCS#1

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

	PSS)		
	DSA	Depends on Algorithm: TEE_ALG_DSA_SHA1 : Between 512 and 1024 bits, multiple of 64 bits TEE_ALG_DSA_SHA224 : 2048 bits TEE_ALG_DSA_SHA256 : 2048 or 3072 bits	FIPS 186-4
	ECDSA	160, 192, 224, 256, 384, 521	FIPS 186-4 ANSI X9.62
	ED25519	256	RFC 8032 RFC 7748
Key Exchange (Shared secret derivation)	DH	From 256 to 2048 bits, multiple of 8 bits.	PKCS #3
	ECDH	192, 224, 256, 384, 521	NIST SP800-56A
	X25519	256	RFC 7748

1.4.3 Reference device life cycle

The TOE is developed as part of the TEE SW design performed during Phase 1, and it will be integrated during Phase 4 of the reference device life cycle described in Figure 2-4 of [GP TEE PP].

1.5 References

This Security Target references the following documents:

Reference	Title
[CAPI]	• TEE Client API Specification, GlobalPlatform (Version 1.0, July 2010, ref: GPD_SPE_007)
[IAPI]	• TEE Internal API Specification, GlobalPlatform (Version v.1.1.2, November 2016, ref: GPD_SPE_010)
[SA]	• TEE System Architecture, GlobalPlatform (Version 1.1, January 2017, ref: GPD_SPE_009)
[KS2011]	• A proposal for Functionality classes for random number generators (Version 2.0, 18 September 2011, part of AIS 20 / 31)
[GP TEE PP]	• TEE Protection Profile, GlobalPlatform (Version 1.2.1, November 2016, ref: GPD_SPE_021)

1.6 Abbreviations used

This document uses the following abbreviations:

Abbreviation	Explanation
CA	Client Application
CC	Common Criteria
CM	Configuration Management
EAL	Evaluation Assurance Level
GP	Global Platform
MMU	Memory Management Unit

OSP	Organisational Security Policy
OTP	One-Time Programmable
PP	Protection Profile
PTRNG	Physical True Random Number Generator
REE	Rich Execution Environment
RNG	Random Number Generator
SAR	Security Assurance Requirement
SFP	Security Function Policy
SFR	Security Functional Requirement
SPD	Security Problem Definition
ST	Security Target
TA	Trusted Application
TSF	TOE Security Functionality
TEE	Trusted Execution Environment
TOE	Target of Evaluation
TSS	TOE Summary Specification

2 CONFORMANCE CLAIMS

2.1 CC Conformance Claim

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

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

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

[1] Common Criteria, Part 1: Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, Version 3.1, Revision 5, April 2017.
[2] Common Criteria, Part 2: Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Components, Version 3.1, Revision , April 2017
[3] Common Criteria, Part 3: Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Components, Version 3.1, Revision 5, April 2017
[4] Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017

2.2 PP Claim

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

2.3 Package Claim

This Security Target claims conformance to the assurance package EAL 2 augmented with AVA_TEE.2 and ALC_FLR.1.

2.4 Conformance Claim Rationale

While the Security Target does not claim conformance to a protection profile, it implements part of the functionality described in the [GP TEE PP]. The CC conformance claim and package claim are in line with the claims made in [GP TEE PP].

The following tables show the relation between the [GP TEE PP] and this ST in terms of the Security Problem Definition, the Security Objectives, and the Security Functional Requirements.

TABLE 3 RELATION OF SPD IN THE [GP TEE PP] AND THIS ST

SPD element	[GP TEE PP]	This ST
T.ABUSE_FUNCT	*	*
T.CLONE	*	*
T.FLASH_DUMP	*	*
T.IMPERSONATION	*	*
T.ROGUE_CODE_EXECUTION	*	*
T.PERTURBATION	*	*
T.RAM	*	*
T.RNG	*	*
T.SPY	*	*
T.TEE_FIRMWARE_DOWNGRADE	*	*
T.STORAGE_CORRUPTION	*	*
OSP.INTEGRATION_CONFIGURATION	*	* (extended to include integration of TOE into TEE)
OSP.SECRETS	*	*
A.PROTECTION_AFTER_DELIVERY	*	*
A.ROLLBACK	*	*
A.TA_DEVELOPMENT	*	*
A.SECURE_INITIALIZATION		*

A.SECURE_PLATFORM		*

TABLE 4 RELATION OF SECURITY OBJECTIVES IN THE [GP TEE PP] AND THIS ST

Security Objective	[GP TEE PP]	This ST
O.CA_TA_IDENTIFICATION	*	*
O.INITIALIZATION	*	Part of this objective has been shifted to OE.INITIALIZATION
O.KEYS_USAGE	*	*
O.OPERATION	*	*
O.RNG	*	The objective has been amended to state that the seed must be provided by the hardware as per OE.SECURE_PLATFORM
O.RUNTIME_CONFIDENTIALITY	*	*
O.RUNTIME_INTEGRITY	*	*
O.TA_AUTHENTICITY	*	*
O.TA_ISOLATION	*	*
O.TEE_DATA_PROTECTION	*	*
O.TEE_ID	*	The objective has been amended to state that the identifier must be provided by the hardware as per OE.SECURE_PLATFORM.
O.TEE_ISOLATION	*	*
O.TRUSTED_STORAGE	*	*
O.INSTANCE_TIME	*	This objective has been shifted to OE.SECURE_PLATFORM
OE.INITIALIZATION		*
OE.INTEGRATION_CONFIGURATION	*	*
OE.PROTECTION_AFTER_DELIVERY	*	*
OE.ROLLBACK	*	*
OE.SECRETS	*	*
OE.SECURE_PLATFORM		*
OE.TA_DEVELOPMENT	*	*

TABLE 5 RELATION OF THE SFRS IN [GP TEE PP] AND THIS ST

SFR	[GP TEE PP]	This ST
Identification		
FIA_ATD.1	*	*
FIA_UID.2	*	*
FIA_USB.1	*	*
FMT_SMR.1	*	*
Confidentiality, Integrity and Isolation		
FDP_IFC.2/Runtime	*	Part of this functionality has to be provided by the platform as per A.SECURE_PLATFORM. It is
FDP_IFF.1/Runtime	*	

		configured correctly by the TOE. *
FDP_ITT.1/Runtime	*	This is provided by the hardware only and the SFR is not included
FDP_RIP.1/Runtime	*	*
FPT_ITT.1/Runtime	*	*
Cryptography		
FCS_COP.1	*	*
FDP_ACC.1/TA_Keys	*	*
FDP_ACF.1/TA_Keys	*	*
FMT_MSA.1/TA_Keys	*	*
FMT_MSA.3/TA_Keys	*	*
Initialization, Operation and Firmware Integrity		
FAU_ARP.1	*	The responsibility is shared between the TOE and the hardware/firmware
FPT_FLS.1	*	
FDP_SDI.2	*	*
FPT_INI.1	*	This SFR has been limited to the implementation-dependent verification checks performed by the TOE itself, as the hardware and firmware have to perform many of the checks described in [GP TEE PP]
FMT_SMF.1	*	*
FPT_TEE.1	*	*
TEE Identification		
FAU_SAR.1	*	The TOE only provides the unique identifier to TA users
FAU_STG.1	*	The TOE shares this responsibility with the hardware.
Instance time		
FPT_STM.1/Instance time	*	Associated objective is shifted to the environment and the SFR is not included
Random Number Generator		
FCS_RNG.1	*	The seed must be provided by the hardware
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	*	*
FDP_ITT.1/Trusted Storage	*	*
Additional SFRs		
FCS_CKM.1		*

FCS_CKM.4		*

3 SECURITY PROBLEM DEFINITION

3.1 Description of Assets

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

TEE identification

TEE identification data 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.

RNG

Random Number Generator.

Properties: unpredictable random numbers, sufficient entropy.

TA code

The code of the installed Trusted Applications. This data is typically stored in external non-volatile memory 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 TEE security services. Data and keys are owned either by the user (the owner of the TEE-enabled device) or by the TA service provider. This data is typically stored in external non-volatile memory 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 TEE.

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.

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. This data is typically stored in the Trusted OTP memory of the TEE.

Properties: integrity and confidentiality.

Application Note:
Confidentiality of this asset is ensured by the simple fact that the asset remains inside the SoC part of the TEE.

3.2 Users / Subjects

There are two kinds of users of the TOE in [GP TEE PP]: Trusted Applications, which use the TOE services through the TEE Internal 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 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.

This Security Target defines an additional user: the system integrator who integrates the software-only TOE into a full TEE solution.

System integrator

The System Integrator, integrating the software-only TOE into a full TEE solution, potentially implementing additional drivers to facilitate the integration.

3.3 Threats

This ST targets threats to the TEE 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 device embedding the TEE. The user of the device becomes a potential attacker when the TEE holds assets of third parties. The motivations behind the attacks may be very diverse and in general are linked to

the Trusted Application running on the TEE. An attacker may, for instance, try to steal content of the device's owner (such as passwords stored in the device) or content of a service provider, or to unduly benefit from TEE or TA services (such as accessing corporate network or performing unauthorized use of DRM content either in the same device or in other devices) or to threaten the reputation of the device/TEE manufacturer or service providers. The impact of an attack depends not only on the value of the individual assets attacked, but, in some cases, on the possibility to reproduce the attack rapidly at low cost: single attacks performed on a given TEE-enabled device have lower impact than massive attacks that reach many devices at the same time.

This ST focuses on non-destructive software attacks that can be easily widespread, for instance through the internet, and constitute a privileged vector for getting undue access to TEE assets without damaging the device itself. In many cases, a software attack involves at least two attackers: the attacker in the identification phase that discovers some vulnerability, conceives malicious software and distributes it, and the attacker at the exploitation phase that effectively exploits the vulnerability by running the malicious software (the end-user or a remote attacker on behalf of the user). The identification and the exploitation attackers may be the same person, in the case of an attack where there is no interest in, or no possibility of spreading the attack widely.

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

Depending on the way the device is used, different assumptions may be valid regarding the identification phase in terms of means available to the attacker, software or hardware, and in terms of possibility to use more than one device, potentially in a destructive way. When identification and exploitation are separate, to address unmanaged devices across which an attack may be easily widespread, the attacker in the identification phase may have software and /or hardware expertise and access to equipment such as oscilloscopes, protocol analyzers, in-circuit emulators, or JTAG debuggers, which allow the attacker to operate at the package interface on the PCB. However, it is not expected that the available attack potential be sufficient to act at deep package and SoC levels

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

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

In all cases, the overall attack potential strongly limits the possibility to face advanced attackers performing the exploits. For large-scale exploitation attacks, we refer to the Annex A of [GP TEE PP] for a comprehensive description of the identification and exploitation phases, the applicable attack potential quotation table and a representative set of attacks a TEE may have to face in the field. Due

to the somehow more limited interest and possibility of spreading the exploits, attacks against managed devices should only be subset of the attacks in Annex A of [GP TEE PP].

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.

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), RNG (confidentiality, integrity), TA code (authenticity, consistency).

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

Application Note:
Attack paths may consist, 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 and TEE data and potentially allowing the attacker to mount other attacks.

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

Application Note:
An attack path consists for instance in performing a (partial) memory dump through the REE, purely via software or with a USB connection. During identification, another example consists 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, RNG.

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, RNG.

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), RNG (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), RNG (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 generated by the TEE. 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): RNG 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 or TA (e.g. timing, power consumption), 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). During the identification phase, the attacker may for instance probe external buses.

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.

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

3.5.1 Assumptions from [GP TEE PP]

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

3.5.1 Assumptions in this ST

Because the TOE in this ST is a software-only solution and some of the security functionality described in [GP TEE PP] relies on the hardware part, several additional assumptions are made regarding the hardware platform that runs this TOE.

A.SECURE_INITIALISATION

It is assumed that the non-TOE hardware and firmware required by the TOE guarantee the secure initialisation of the TOE as mandated by [GP TEE PP], including:

- the integrity of the full software TOE as described in this ST
- the integrity of TEE initialization code and data
- the authenticity and integrity of TEE firmware
- the integrity of the storage root of trust
- the integrity of the TEE identification data
- the version of the firmware to prevent downgrade to previous versions

A.SECURE_PLATFORM

- protects internal data transfer between physically separated components of the TOE;
- provides functionality to isolate the TOE from the REE, as well as to isolate different entities within the TOE;
- prevents an attacker from abusing physical implementation characteristics of the hardware to compromise TOE assets via side-channel analysis (e.g. power consumption, EM emanation, etc.);
- prevents an attacker from compromising TOE assets via perturbation or physical attacks;
- provides a device-unique key derived from the globally unique device id and globally unique device key, which is bound to the globally unique TEE ID as the TEE ID is derived from the globally unique device id;
- provides some rollback-protected memory, a timer, and an entropy source of sufficient quality (i.e., at least satisfying the requirements for PTG.2 as described in [KS2011]).

4 SECURITY OBJECTIVES

4.1 Security Objectives for the TOE

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

O.CA_TA_IDENTIFICATION

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

O.KEYS_USAGE

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

O.TEE_ID

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

Application Note:
The hardware shall provide the unique identifier as per OE.SECURE_PLATFORM.

O.INITIALIZATION

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

- All hardware peripherals required by the TOE are available.
- All services of the TOE have correctly booted.

Application Note:
The integrity and authenticity of the non-TOE firmware, as well as the integrity and authenticity of the TOE itself is guaranteed by OE.INITIALIZATION

O.OPERATION

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

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

Application Note:

- Programmer errors or violation of good practices (e.g. that exploit multi-threading or context/session management) might become attack-enablers. The REE may be harmful but «the implementation (TEE) still guarantees the stability and security of TEE » (cf. [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 (cf. [IAPI] and [SA])
- Entry points (cf. [SA]): Software in the REE must not be able to call directly to TEE Functions or Trusted Core Framework. The REE software must go through protocols such that the Trusted OS or Trusted Application performs the verification of the acceptability of the operation that the REE software has requested.

O.RNG

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

Application Note:
Random number generation in the TOE is based on a deterministic hybrid random number generator that must be seeded with sufficient entropy from a hardware-based entropy source as per OE.SECURE_HARDWARE_PLATFORM.

O.RUNTIME_CONFIDENTIALITY

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

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

O.RUNTIME_INTEGRITY

The TEE shall ensure that the TEE firmware, 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 TEE shall verify code authenticity of Trusted Applications.

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

Application Note:
This objective contributes to the enforcement of the confidentiality and the integrity of the TEE.

O.TEE_DATA_PROTECTION

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

O.TEE_ISOLATION

The TEE shall prevent the REE and the TAs from accessing the TEE 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 TEE 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 TEE 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 TEE 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.

TABLE 6

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

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.SECURE_PLATFORM.

4.2 Security Objectives for the environment

This section states the security objectives for the TEE 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.

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 TEE 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, prepersonalization 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.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)
- 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.

OE.INITIALIZATION

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

- the integrity of the full software TOE as described in this ST
- the integrity of TEE initialization code and data
- the authenticity and integrity of TEE firmware
- the integrity of the storage root of trust
- the integrity of the TEE identification data
- the version of the firmware to prevent downgrade to previous versions

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 ([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.SECURE_PLATFORM

The platform running the TOE shall consist of secure hardware and firmware that provide features allowing the TOE to implement secure functionality.

The security functions supported/provided by the trusted platform are:

- A globally unique and non-modifiable TEE identifier that is stored in secure OTP memory
- A timer to provide reliable timestamps
- An entropy source of sufficient quality to seed the RNG of the TOE (i.e., at least satisfying the requirements for PTG.2 as described in [KS2011])
- A security mechanism enabling the TOE to isolate the REE and the TEE side
- A security mechanism enabling the TOE to isolate different internal processes
- A device-unique key derived from the globally unique TEE ID that is made available to the TEE side.
- Protection of the memories against tampering.
- A memory that is protected against rollback.
- Protection from side-channel attacks that exploit physical implementation characteristics of the hardware to compromise TOE assets stored and/or handled by the platform.
- Protection from physical and perturbation attacks that compromise TOE assets stored and/or handled by the platform.

Application Note:
For this TOE, the fourth point is satisfied by Arm TrustZone technology, whereas the fifth point is satisfied by a configurable hardware MMU and privilege levels.

4.3 Security Objectives Rationale

4.3.1 Threats

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

- O.INITIALIZATION and 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) with the support of OE.SECURE_PLATFORM
- O.KEYS_USAGE controls the usage of cryptographic keys
- OE.TA_DEVELOPMENT enforces TA development principles, which are meant in particular to prevent disclosing information or performing modifications upon request of unauthorized entities.

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

- O.TEE_ID and OE.SECURE_PLATFORM provide the unique TEE identification means
- O.INITIALIZATION and 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
- O.RNG ensures (supported by OE.SECURE_PLATFORM) that the TEE identifier is in fact unique when generated inside the TOE

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

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

- O.CA_TA_IDENTIFICATION ensures the protection of Client identities and the possibility to distinguish Client Applications and Trusted Applications
- O.OPERATION ensures the verification of Client identities before any operation on their behalf
- O.RUNTIME_INTEGRITY prevents against unauthorized modification of security functionality at runtime.

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

- O.INITIALIZATION and 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:

- O.INITIALIZATION and 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).

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

- O.INITIALIZATION and 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.

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

- O.INITIALIZATION and OE.INITIALIZATION ensure the correct initialization of the TEE security functions, in particular the RNG
- O.RNG ensures (supported by OE.SECURE_PLATFORM) that random numbers are unpredictable, have sufficient entropy and are not disclosed
- O.RUNTIME_CONFIDENTIALITY ensures that confidential data is not disclosed
- O.RUNTIME_INTEGRITY protects against unauthorized modification, for instance to force the output of the RNG.

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

- O.RUNTIME_CONFIDENTIALITY ensures protection of confidential data at runtime
- O.TA_ISOLATION ensures the separation between TAs
- O.TEE_ISOLATION ensures that neither REE nor TAs can access TEE data
- O.TRUSTED_STORAGE ensures that data stored in the trusted storage locations is accessible by the TA owner only.

T.TEE_FIRMWARE_DOWNGRADE The combination of the following objectives ensures protection against TEE firmware 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
- O.INITIALIZATION and OE.INITIALIZATION ensure that the firmware that is executed is the version that was intended
- OE.ROLLBACK states the limits of the properties enforced by the TSF.

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

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

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

4.3.4 Coverage

TABLE 7 MAPPING OF OBJECTIVES TO THREATS, OSPS, OR ASSUMPTIONS

Threat, OSP, or Assumption	Objectives
T.ABUSE_FUNCT	O.INITIALIZATION, OE.INITIALIZATION, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_AUTHENTICITY, O.TEE_DATA_PROTECTION, O.TEE_ISOLATION, OE.SECURE_PLATFORM, O.KEYS_USAGE, OE.TA_DEVELOPMENT
T.CLONE	O.TEE_ID, O.INITIALIZATION, OE.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, O.RNG, OE.SECURE_PLATFORM
T.FLASH_DUMP	O.TRUSTED_STORAGE
T.IMPERSONATION	O.CA_TA_IDENTIFICATION O.OPERATION O.RUNTIME_INTEGRITY
T.ROGUE_CODE_EXECUTION	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION,

	O.TRUSTED_STORAGE, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY, O.TA_AUTHENTICITY, O.INITIALIZATION, OE.INITIALIZATION
T.PERTURBATION	O.INITIALIZATION, OE.INITIALIZATION, OE.INSTANCE_TIME, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_DATA_PROTECTION, O.TEE_ISOLATION, O.TA_AUTHENTICITY
T.RAM	O.INITIALIZATION, OE.INITIALIZATION O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION
T.RNG	O.INITIALIZATION, OE.INITIALIZATION, O.RNG, OE.SECURE_PLATFORM, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY
T.SPY	O.RUNTIME_CONFIDENTIALITY, O.TA_ISOLATION, O.TEE_ISOLATION, O.TRUSTED_STORAGE
T.TEE_FIRMWARE_DOWNGRADE	O.INITIALIZATION, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY
T.STORAGE_CORRUPTION	O.OPERATION, OE.ROLLBACK, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, O.TA_AUTHENTICITY, O.INITIALIZATION, OE.INITIALIZATION
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.SECURE_INITIALIZATION	OE.INITIALIZATION
A.SECURE_PLATFORM	OE.SECURE_PLATFORM

TABLE 8 MAPPING OF THREATS, OSPS, OR ASSUMPTIONS TO OBJECTIVES

Objective	Threat, OSP, or Assumption
O.CA_TA_IDENTIFICATION	T.IMPERSONATION
O.INITIALIZATION	T.CLONE T.ABUSE_FUNCT T.ROGUE_CODE_EXECUTION T.PERTURBATION T.RAM T.RNG T.TEE_FIRMWARE_DOWNGRADE T.STORAGE_CORRUPTION
O.KEYS_USAGE	T.ABUSE_FUNCT
O.OPERATION	T.ABUSE_FUNCT T.IMPERSONATION T.ROGUE_CODE_EXECUTION T.PERTURBATION T.STORAGE_CORRUPTION
O.RNG	T.CLONE T.RNG
O.RUNTIME_CONFIDENTIALITY	T.ABUSE_FUNCT T.CLONE

	T.ROGUE_CODE_EXECUTION T.PERTURBATION T.RAM T.RNG T.SPY
O.RUNTIME_INTEGRITY	T.ABUSE_FUNCT T.CLONE T.IMPERSONATION T.ROGUE_CODE_EXECUTION T.PERTURBATION T.RAM T.RNG
O.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.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.SECRETS	OSP.SECRETS
OE.SECURE_PLATFORM	T.ABUSE_FUNCT

	T.CLONE T.RNG A.SECURE_PLATFORM
OE.TA_DEVELOPMENT	T.ABUSE_FUNCT A.TA_DEVELOPMENT
O.TEE_ID	T.CLONE

5. EXTENDED COMPONENTS DEFINITION

5.1 Extended family FCS_RNG

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

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

It contains the single component FCS_RNG.1.

5.1.1 Extended component FCS_RNG.1

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

Hierarchical to: No other components.

Management: No management activities are foreseen.

Audit: No actions are defined to be auditable.

FCS_RNG.1 Random numbers generation

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

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

Dependencies: No dependencies

5.2 Extended family FPT_INI

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

It contains the single component FPT_INI.1.

5.2.1 Extended component FPT_INI.1

Requirement for the TOE to provide a TSF initialization function that brings the TSF into a secure operational state at power-on.

Hierarchical to: No other components.

Management: No management activities are foreseen.

Audit: No actions are defined to be auditable.

FPT_INI.1 TSF initialisation

FPT_INI.1.1 The TOE initialization function shall verify [assignment: list of verifications] prior to establishing the TSF in a secure initial state.

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

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

Dependencies: No dependencies.

Note that this extended component definition slightly differs from that of [GP TEE PP], which includes TEE-specific content in the extended component definition, which is not in line with component definitions as described in CC Part 2.

5.3 Extended family AVA_TEE

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

The potential vulnerabilities may be identified either during the evaluation of the development, manufacturing or assembling environments, during the evaluation of the TEE specifications and guidance, during anticipated operation of the TEE components or by other methods, for instance statistical methods.

The family 'Vulnerability analysis of TEE (AVA_TEE)' defines requirements for evaluator independent vulnerability search and penetration testing of TEE.

The main characteristic of the new family, which is almost identical to AVA_VAN, is to introduce the TEE-specific attack potential scale defined in Annex A.1. This annex also provides the TEE attack potential calculation table and a catalogue of TEE-specific attack methods. In this current version of the Protection Profile, only one level of the TEE-specific attack potential scale, namely TEE-Low, is used, in the component AVA_TEE.2.

By comparison with the family AVA_VAN, the standard component AVA_VAN.2 of this family provides a good level of assurance against SW-only attacks on TEE, for instance mobile application malware that is spreading through uncontrolled application stores, exploiting already known SW vulnerabilities. Such malwares can usually defeat REE security, and the TEE must resist such attacks. AVA_VAN.2 is well-fit for devices managed within a controlled environment, e.g. fleets of corporate devices, for services against which the end-user may not have any interest in attacking.

This choice of a TEE-specific attack potential scale in AVA_TEE.2 is motivated by additional assurance with protection against easily spreadable attacks that may result from costly vulnerability identification. Such attack paths have been used in some cases against mobile devices, and are usual in market segments such as game consoles or TV boxes, where the expected return on investment is higher, and in which the end-user has an interest to perform the exploit. In order to reach this goal, AVA_TEE.2 splits attacks quotation in the two phases identification and exploitation (as done for smart cards) and defines the attacker potential TEE-Low (which is comparable to the Enhanced-Basic level of the smart cards quotation table).

The family 'Vulnerability analysis of TEE (AVA_TEE)' contains the single extended component AVA_TEE.2, which is defined as follows. Underlined text stands for replacements with respect to AVA_VAN.2, thus allowing easy traceability.

5.3.1 Extended component AVA_TEE.2

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.

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. Operations in SFRs and SARs are described in bold font.

Users stand for entities outside the TOE:

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

Subjects stand for active entities inside the TOE:

- S.TA_INSTANCE: any TA instance with security attribute "TA_identity" (TA identifier)
- S.TA_INSTANCE_SESSION: any session within a given TA instance, with security attribute "client_identity" (CA identifier)
- S.API: the TEE Internal 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 (cf. FDP_ITT.1)
- S.RAM_UNIT: RAM addressable unit, with security attribute "rights:(TA identifier/REE) ->(Read/Write/ReadWrite/NoAccess). For instance, an addressable unit may be allocated or have its access rights changed upon TA instance creation or when sharing memory references between a client (CA, TA) and a TA. Notes: 1) A RAM_UNIT typically stands for a byte in the C language; 2) there is no RAM access restriction applicable to the TEE itself
- S.COMM_AGENT: proxy between CAs in the REE and the TEE and its TAs.

Objects stand for passive entities inside the TOE:

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

Cryptographic objects are a special kind of TEE objects:

- OB.TA_KEY (user data): (handle to a) user key (persistent or transient), with security attributes "usage", "owner" (TA identifier), "isExtractable" (True/False).
- Information stands for data exchanged between subjects:
- I.RUNTIME_DATA (user data or TSF Data depending on the owner): data belonging to the TA or to the TEE itself. Stands for parameter values, return values, content of memory regions in cleartext. Note: Data that is encrypted and authenticated is not considered I.RUNTIME_DATA.

TSF data consists of runtime 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.
Cryptographic operations on user keys performed by S.API on behalf of TA_INSTANCE:

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

Trusted Storage operations performed by S.API on behalf of TA_INSTANCE:

- OP.LOAD: any operation used to get back persistent objects (data and keys) to be used by the TA
- OP.STORE: any operation used to store persistent objects (data and keys). It stands for object creation, object deletion, object renaming, object truncation and write to an object.

Other operations:

- Any operation executed by the TEE on behalf of a TA_INSTANCE.

This 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, FDP_ITT.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 intances: 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.

6.1 Security Functional Requirements

6.1.1 BASE 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.**

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**.

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**

- **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**.

Application Note:
TEE Internal API defines the codification rules of the CA identity.

FMT_SMR.1 Security roles
FMT_SMR.1.1 The TSF shall maintain the roles
- **TSF**
- **TA_User**

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

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

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.

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

Application Note:

- The access rights configuration managed by S.RAM_UNIT shall ensure that RAM addressable units used to TSF data are appropriately protected (in integrity for TEE firmware, 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: **TEE and TA runtime objects**.

Application Note:
This operation applies in particular upon:
• Failure detection (cf. FPT_FLS.1)
• TA instance and TA session closing.

FPT_ITT.1/Runtime Basic internal TSF data transfer protection
FPT_ITT.1.1/Runtime The TSF shall protect TSF data from **disclosure and modification** when it is transmitted between separate parts of the TOE.

Application Note:
The resources used by the TEE may reside in "physically separated parts".
The TOE implements this SFR based on the hardware TrustZone mechanism.

6.1.1.3 Cryptography

FCS_COP.1 Cryptographic operation
FCS_COP.1.1 The TSF shall perform **list of cryptographic operations in Table 2** in accordance with a specified cryptographic algorithm **listed in Table 2** and cryptographic key sizes **listed in Table 2** that meet the following: **corresponding list of standards in Table 2**.

Application Note:
TA code is verified using an RSA-2048 signature and encrypted 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. Other operations are provided as an API.

FDP_ACC.1/TA_keys Subset access control
FDP_ACC.1.1/TA_keys The TSF shall enforce the **TA Keys Access Control SFP** on

- **Subjects: S.API, S.TA_INSTANCE and any other subject in the TEE**
- **Objects: OB.TA_KEY**
- **Operations: OP.USE_KEY, OP.EXTRACT_KEY**.

FDP_ACF.1/TA_keys Security attribute based access control

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

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

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

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

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

- **Any access to a user key attempted directly from S.TA_INSTANCE or any other subject of the TEE that is not S.API**
- **Any access to a user key attempted from S.API without valid caller (S.API.caller is undefined)**

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

FMT_MSA.1/TA_keys Management of security attributes
FMT_MSA.1.1/TA_keys The TSF shall enforce the **TA Keys Access Control SFP** to restrict the ability to **change_default, query and modify** the security attributes **OB.TA_KEY.usage, OB.TA_KEYS.isExtractable and OB.TA_KEY.owner** to **the following roles:**

- **change_default, query and modify OB.TA_KEY.usage to TA_User role**
- **query OB.TA_KEY.owner to the TSF role**.

FMT_MSA.3/TA_keys Static attribute 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** 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 action of halting of the TSF, aborting the execution of the TA instance, or returning an error** upon detection of a potential security violation.

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

- **detection of consistency violation of TA data, TA code or TEE data: return an error in case of TA data, abort the execution of the TA instance in case of TA code, halt the TSF in case of TEE data.**
- **detection of TEE firmware integrity violation: halt the TSF.**

FDP_SDI.2 Stored data integrity monitoring and action
FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for **authenticity and consistency errors** on all objects, based on the following attributes: **signature of TEE persistent data, access to TEE runtime data, MAC on TA data and keys and signature of TA code**.

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

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

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **halt the TSF, abort the execution of the TA instance, or return an error**.
Refinement:
- **Upon detection of authenticity or consistency errors in TEE persistent data, the TSF shall halt the TSF**
- **Upon detection of an access control violation to the TEE runtime data, the TSF shall deny access**
- **Upon detection of TA code authenticity or consistency errors, the TSF shall abort the execution of the TA instance**
- **Upon detection of TA data or TA keys authenticity or consistency errors, the TSF shall**
    - **Not give back any compromised data**
    - **Return an error independent of the compromised data**

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**
- **TEE data (in particular TA properties, TEE keys and all security attributes) authenticity or consistency failure**
- **TEE initialization failure**
- **Unexpected commands in the current TEE state**

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.

FPT_INI.1 TSF initialisation
FPT_INI.1.1 The TOE initialization function shall verify
- **The availability of hardware peripherals**
- **The correct initialization of TOE services**

prior to establishing the TSF in a secure initial state.

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

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

Application Note:
For the other verifications described in [GP TEE PP], the TOE relies on the underlying platform.

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** 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.

Application Note:
As per OE.SECURE_PLATFORM, the globally unique TEE identifier must be provided by the hardware, and it must be stored in secure OTP memory or derived from a value stored in secure OTP memory. As described in the conformance claim, this Security Target does not conform to any Protection Profile. If a user wishes to integrate the TOE of this Security Target into a TEE that is compliant to [GP TEE PP], then the user of the TOE must implement a TA that allows all users to access the TEE identifier.

FAU_STG.1 Protected audit trail storage
FAU_STG.1.1 The TSF shall protect the stored audit records in the audit trail from unauthorized deletion.

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

6.1.1.6 Random Number Generator

FCS_RNG.1 Random numbers generation
FCS_RNG.1.1 The TSF shall provide a **deterministic** random number generator that implements:
(DRG.2.1) If initialized with a random seed using a PTRNG of class PTG.2 as random source, the internal state of the RNG shall have at least 200 bits of entropy.
(DRG.2.2) The RNG provides forward secrecy.
(DRG.2.3) The RNG provides backward secrecy.

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

(DRG.2.4) The RNG, initialized with a random seed using a PTRNG of class PTG.2 as random source, generates output for which 2^{35} strings of bit length 128 are mutually different with probability $1-2^{-17}$.
(DRG.2.5) Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.

Application Note:
As per OE.SECURE_PLATFORM, the PTRNG that provides the random seed for initialization must be provided by the environment. For a definition of the above terms and the requirements of a PTG.2 PTRNG, see [KS2011].

6.1.1.7 Trusted Storage

FDP_ACC.1/Trusted Storage Subset access control
FDP_ACC.1.1/Trusted Storage The TSF shall enforce the **Trusted Storage Access Control SFP** on

- **Subjects: S.API**
- **Objects: OB.TA_STORAGE, OB.SRT**
- **Operations: OP.LOAD, OP.STORE**.

FDP_ACF.1/Trusted Storage Security attribute based access control
FDP_ACF.1.1/Trusted Storage The TSF shall enforce the **Trusted Storage Access Control SFP** to objects based on the following: **S.API.caller, OB.TA_STORAGE.owner, OB.TA_STORAGE.inExtMem, OB.TA_STORAGE.TEE_identity and OB.SRT.TEE_identity**.

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

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

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

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

- **Any access to a trusted storage attempted from S.API without valid caller (S.API.caller = undefined)**
- **Any access to a trusted storage that was bound to a different TEE (OB.TA_STORAGE.TEE_identity different from OB.SRT.TEE_identity)**
- **Any access to a trusted storage from a subject different from S.API**

FDP_ROL.1/Trusted Storage Basic rollback
FDP_ROL.1.1/Trusted Storage The TSF shall enforce **Trusted Storage Access Control SFP** to permit the rollback of the **unsuccessful or interrupted OP.STORE operation** on the **storage**.

FDP_ROL.1.2/Trusted Storage The TSF shall permit operations to be rolled back within the **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.

6.1.4 SFRs additional to the PP

FCS_CKM.1 Cryptographic key generation
FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **key generation for the cryptographic algorithms listed in Table 2** and specified cryptographic key sizes **listed in Table 2** that meet the following: **corresponding list of standards in Table 2 and [IAPI]**.

Application Note:
The key derivation performed for secure storage is done by the environment under as part of OE.SECURE_PLATFORM.

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 **overwriting with a fixed value** that meets the following: **none**.

6.2 Security Assurance Requirements

The Security Target will be evaluated according to

Security Target evaluation (Class ASE)

The TOE claims Evaluation Assurance Level EAL2 augmented with AVA_TEE.2 and ALC_FLR.1. The overview below lists the assurance components claimed by the TOE.

Class ADV: Development

Security architecture description	(ADV_ARC.1)
Security-enforcing functional specification	(ADV_FSP.2)
Basic design	(ADV_TDS.1)

Class AGD: Guidance documents activities

Operational User Guidance	(AGD_OPE.1)
Preparative procedures	(AGD_PRE.1)

Class ALC: Life-cycle support

CM Capabilities	(ALC_CMC.2)
CM Scope	(ALC_CMS.2)
Delivery	(ALC_DEL.1)
Basic flaw remediation	(ALC_FLR.1)

Class ASE: Security Target evaluation

Conformance claims	(ASE_CCL.1)
Extended components definition	(ASE_ECD.1)
ST introduction	(ASE_INT.1)
Security objectives	(ASE_OBJ.2)
Derived security requirements	(ASE_REQ.2)
Security problem definition	(ASE_SPD.1)
TOE summary specification	(ASE_TSS.1)

Class ATE: Tests

Evidence of coverage	(ATE_COV.1)
Functional testing	(ATE_FUN.1)
Independent Testing - Sample	(ATE_IND.2)

Class AVA: Vulnerability Assessment

Vulnerability analysis	(AVA_VAN.2)
TEE Vulnerability Analysis	(AVA_TEE.2)

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 specifies the allowed operations
- FCS_CKM.1 specifies 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.
- FAU_STG.1 prevents modification of the unique identifier.

O.INITIALIZATION The following requirements contribute to fulfill the objective:
- FPT_FLS.1 states that the TEE has to reach a secure state upon initialization or device binding failure
- FPT_INI.1 enforces the initialization of the TSF through a secure process of verifications

O.OPERATION The following requirements contribute to fulfill the objective:
- FAU_ARP.1 states the TEE responses to potential security violations
- FDP_SDI.2 enforces the monitoring of consistency and authenticity of TEE data and TA, and it states the 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 TEE
- FPT_FLS.1 states that abnormal operations have to lead to a secure state
- FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage,
- FMT_MSA.3/Trusted Storage and FMT_SMF.1 state the policy for controlling access to TA storage
- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state the policy for controlling access to TA and TEE execution spaces
- FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys and FMT_SMF.1 state the key access policy.

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

O.RUNTIME_CONFIDENTIALITY The following requirements contribute to fulfill the objective:
- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime ensure read access to authorized entities only
- FPT_ITT.1/Runtime ensures protection against disclosure of TEE and TA data that is transferred between resources
- FDP_RIP.1/Runtime states resource clean up policy.

O.RUNTIME_INTEGRITY The following requirements contribute to fulfill the objective:
- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state TEE and TA runtime data policy, which grants write access to authorized entities only
- FPT_ITT.1/Runtime ensures protection against modification of TEE and TA data that is transferred between resources
- FDP_SDI.2 monitors the authenticity and consistency of TEE code, the TEE runtime data, the TA code and the TA data and keys and states the response upon failure.

O.TA_AUTHENTICITY The following requirements contribute to 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 states the cryptography used to verify the authenticity of TA code

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 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 states the cryptography used to protect consistency and confidentiality of the TEE data in external memory
- FDP_SDI.2 monitors the authenticity and consistency of TEE persistent data and states the response upon failure
- FPT_ITT.1/Runtime enforces secure transmission and storage of TEE persistent data.

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 TEE execution space.

O.TRUSTED_STORAGE The following requirements contribute to fulfill the objective:
- FCS_COP.1 states the cryptography used to protect integrity and confidentiality of the 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

- FPT_INI.1 enforces the integrity of TEE identification and storage root of trust, and it states the behavior in case of failure
- FPT_FLS.1 maintains a secure state.

TABLE 9 MAPPING OF SFRS TO TOE OBJECTIVES

Objective	SFRs
O.CA_TA_IDENTIFICATION	FIA_ATD.1 FIA_UID.2 FIA_USB.1
O.KEYS_USAGE	FCS_COP.1 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 FCS_CKM.4
O.TEE_ID	FAU_SAR.1 FAU_STG.1
O.INITIALIZATION	FPT_FLS.1 FPT_INI.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.RNG	FCS_RNG.1
O.RUNTIME_CONFIDENTIALITY	FDP_IFC.2/Runtime FDP_IFF.1/Runtime FPT_ITT.1/Runtime FDP_RIP.1/Runtime
O.RUNTIME_INTEGRITY	FDP_IFC.2/Runtime FDP_IFF.1/Runtime FPT_ITT.1/Runtime FDP_SDI.2
O.TA_AUTHENTICITY	FDP_SDI.2 FPT_TEE.1 FCS_COP.1
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 FDP_IFC.2/Runtime

	FDP_IFF.1/Runtime FPT_FLS.1
O.TEE_DATA_PROTECTION	FCS_COP.1 FDP_SDI.2 FPT_ITT.1/Runtime
O.TEE_ISOLATION	FDP_IFC.2/Runtime FDP_IFF.1/Runtime
O.TRUSTED_STORAGE	FCS_COP.1 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.2 FPT_INI.1 FPT_FLS.1

TABLE 10 MAPPING OF TOE OBJECTIVES TO SFRS

SFR	Objectives
FAU_ARP.1	O.OPERATION
FAU_SAR.1	O.TEE_ID
FAU_STG.1	O.TEE_ID
FCS_COP.1	O.KEYS_USAGE O.TA_AUTHENTICITY O.TA_ISOLATION O.TEE_DATA_PROTECTION O.TRUSTED_STORAGE
FCS_CKM.1	O.KEYS_USAGE
FCS_CKM.4	O.KEYS_USAGE
FCS_RNG.1	O.RNG
FDP_ACC.1/TA_keys	O.KEYS_USAGE O.OPERATION
FDP_ACC.1/Trusted Storage	O.OPERATION O.TA_ISOLATION O.TRUSTED_STORAGE
FDP_ACF.1/TA_keys	O.KEYS_USAGE O.OPERATION
FDP_ACF.1/Trusted Storage	O.OPERATION O.TA_ISOLATION O.TRUSTED_STORAGE
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 O.RUNTIME_CONFIDENTIALITY
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.TEE_DATA_PROTECTION 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.INITIALIZATION O.OPERATION O.TA_ISOLATION O.TRUSTED_STORAGE
FPT_INI.1	O.INITIALIZATION O.TRUSTED_STORAGE
FPT_ITT.1/Runtime	O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY O.TEE_DATA_PROTECTION
FPT_TEE.1	O.TA_AUTHENTICITY

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.

TABLE 11 SFR DEPENDENCY ANALYSIS

Security Functional Requirement	Dependencies	Fulfilled by security requirements
FIA_ATD.1	No Dependencies	
FIA_UID.2	No Dependencies	
FIA_USB.1	(FIA_ATD.1)	FIA_ATD.1
FMT_SMR.1	(FIA_UID.1)	FIA_UID.2
FDP_IFC.2/Runtime	(FDP_IFF.1)	FDP_IFF.1/Runtime
FDP_IFF.1/Runtime	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/Runtime (See below)
FDP_RIP.1/Runtime	No Dependencies	
FPT_ITT.1/Runtime	No Dependencies	
FCS_COP.1	(FCS_CKM.1 or FDP_ITC.1 or	FCS_CKM.1, FCS_CKM.4

	FDP_ITC.2) and (FCS_CKM.4)	(See below)
FCS_CKM.1	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_CKM.4, FCS_COP.1
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	FCS_CKM.1
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)	FMT_SMR.1, FDP_ACC.1/TA_keys, FMT_SMF.1
FMT_MSA.3/TA_keys	(FMT_MSA.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_MSA.1/TA_keys
FAU_ARP.1	(FAU_SAA.1)	(See below)
FDP_SDI.2	No Dependencies	
FPT_FLS.1	No Dependencies	
FPT_INI.1	No Dependencies	
FMT_SMF.1	No Dependencies	
FPT_TEE.1	No Dependencies	
FAU_SAR.1	(FAU_GEN.1)	(See below)
FAU_STG.1	(FAU_GEN.1)	(See below)
FCS_RNG.1	No Dependencies	
FDP_ACC.1/Trusted Storage	(FDP_ACF.1)	FDP_ACF.1/Trusted Storage
FDP_ACF.1/Trusted Storage	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/Trusted Storage, FMT_MSA.3/Trusted Storage
FDP_ROL.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted Storage
FMT_MSA.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_SMF.1, FDP_ACC.1/Trusted Storage
FMT_MSA.3/Trusted Storage	(FMT_MSA.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_MSA.1/Trusted Storage
FDP_ITT.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted Storage

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

The dependency FCS_CKM.4 of FCS_COP.1 is discarded. The TEE storage root of trust used for cryptographic operations in FCS_COP.1 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.

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.

The dependency FAU_GEN.1 of FAU_SAR.1 and FAU_STG.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.

6.3.3 Rationale for the Assurance Requirements

The Security Assurance Requirements are in line with the certified Protection Profile [GP TEE PP] augmented with ALC_FLR.1. Therefore, they are suitable for this TOE, which implements part of the functionality described in that PP. The addition of ALC_FLR.1 gives additional assurance that security flaws reported in the TOE will be remedied.

6.3.4 Dependencies of Security Assurance Requirements

The dependencies of the Security Assurance Requirements are met because EAL2 is a consistent package, and because the dependencies of the augmentations are satisfied by EAL2: ALC_FLR.1 has no dependencies, and the dependencies of AVA_TEE.2 (ADV_ARC.1, ADV_FSP.2 , ADV_TDS.1, AGD_OPE.1, and AGD_PRE.1) are met by EAL2.

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, i.e., FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FPT_ITT.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 (FDP_ITT.1/Runtime). 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. 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, i.e., FIA_ATD.1, FIA_UID.2, FIA_USB.1, and FMT_SMR.1.

7.3 Trusted storage of TA and TEE data and keys

The Trusted Storage server is a service running inside the TEE, which implements the SFRs related to 6.1.1.7 Trusted Storage, i.e., 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. The trusted Storage Server guarantees authenticity, integrity, and confidentiality by implementing appropriate cryptographic functionality (cf. FCS_COP.1) 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. The unique TEE identifier can be retrieved by the TAs, which implements the SFRs related to 6.1.1.5 TEE Identification, i.e., FAU_SAR.1 and FAU_STG.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, i.e., 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 6.1.1.7 Trusted

Storage, i.e., FDP_ROL.1. The unique identifier (FAU_SAR.1) is provided by the hardware platform which is stored in OTP (one time programmable) memory and is accessible via API exposed to TAs.

7.4 Random Number Generator

By implementing a Hybrid Deterministic RNG, FCS_RNG.1 as described in 6.1.1.6 Random Number Generator is implemented. Note that a hardware seed needs to be provided.

7.5 Cryptographic API

By implementing the cryptographic functionality described in the [IAPI] in addition to the cryptographic functionality used to secure the other TOE functions, the TOE implements the SFRs related to 6.1.1.3 Cryptography, i.e., FCS_COP.1, 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 6.1.4 SFRs additional to the PP, i.e., FCS_CKM.1 and FCS_CKM.4.

7.6 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, i.e., 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) coupled with unique identifiers for each TA. Additionally, TA code is encrypted. Apart from that the authenticity and consistency of TEE code, and TEE runtime data is achieved by the environment (OE.INITIALIZATION): the secure bootloader and the hardware platform.

7.7 Correct execution of TEE

The correct execution of the TEE is achieved by implementing secure initialisation procedures and various verifications at runtime. Firstly, after power-up the secure bootloader of the hardware platform checks the integrity and authenticity of the TOE image (OE.INITALIZATION, FDP_SDI.2). This includes also verification of authenticity and consistency of TEE data which is part of the TOE image. After successful verification by the OE.INITIALIZATION, the further initialization process is performed by the TOE itself. During that process a failure of 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 FAU_ARP.1). During the 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. 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, i.e., FPT_FLS.1, FAU_ARP.1, FDP_SDI.2, and FPT_INI.1.