Unisoc TEE OS Security Target Lite

Date: 2023-11-06

Created by

Change History

Version	Date	Author	Comment
0.30	2023/11/06	Jtsec	Final version

Table of contents

1 ST INTRODUCTION

1.1 ST REFERENCE

Title: Unisoc TEE OS Security Target Lite

Version: v0.30

Author: Unisoc Technologies Co., Ltd.

Evaluation Lab: Riscure

Date of publication: 2023-11-06

This Security Target is for the evaluation of the Unisoc TEE OS, which is a simplified Real-time Operating System aiming to provide the Trusted Execution Environment for managing digital copyright and protecting mobile payment and sensitive data. The TOE version is provided in Section 1.2.

1.2 TOE REFERENCE

TOE Name: Unisoc TEE OS

TOE Developer: Unisoc Technologies Co., Ltd.

TOE Version: v2.1.2

1.3 TOE OVERVIEW

1.3.1 INTRODUCTION

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

The TOE is for embedded devices implementing GlobalPlatform TEE specifications. Even though this ST does not claim conformance with any PP, it is built using items from **[GPD_SPE_021]** (Section 2). The Security Problem Definition as well as the Security Requirements have been adapted, as the TOE comprises only the software part of the PP.

1.3.2 TOE TYPE

The TOE type is the TOS, which is only the software part of the Trusted Execution Environment (TEE) defined by **[GPD_SPE_021]**. A TEE is an execution environment isolated from any other execution environment, including the usual Regular Execution Environment (REE) and their applications.

The TOE is the TOS of the Trusted Execution Environment (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, and time management and arithmetical APIs.

For this TEE, the optional TEE Time and Rollback and TEE Debug modules of **[GPD_SPE_021]** are not implemented. Only the Core TEE Protection Profile has been implemented.

1.3.3 TOE USAGE & MAJOR SECURITY FEATURES

1.3.3.1 TOE SECURITY FEATURES

The TOE security functionality which is in the scope of the evaluation consists of the following features that are described in depth in Section 1.4.1.

- Protection of TOE Security Functionality (TSF)
- Trusted Storage
- Cryptographic operations with functional dependency on the Hardware (non-TOE)
- User Identification and Authentication
- Protection of Trusted Applications (TA).
- Protected communication between Client Applications (CA) and Trusted Applications (TA).

1.3.3.2 TOE INTENDED USAGE

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

- Corporate services: Enterprise devices that enable push e mail access and office applications give employees a flexibility that requires a secure and fast link to their workplace applications through secure storage of their data, remote management of the device by the IT department, or enhancement of security in web-related scenarios.
- Content management: Today's devices offer HD video playback and streaming, mobile TV broadcast reception, and console-quality 3-D games. This functionality often requires content protection, through Digital Rights Management (DRM) or Conditional Access.
- Personal data protection: Device's store increasing amounts of personal information (such as contacts, messages, photos and video clips) and even sensitive data (credentials, passwords, health data, etc.). Secure storage means are required to prevent exposure of this information in the event of loss, theft, or any other adverse event, such as a malware.

- Connectivity protection: Networking through multiple technologies—such as 3G, 4G or Wi-Fi/WiMAX, as well as personal communication means, such as Bluetooth® and Near Field Communication (NFC) — enables the use of mobile devices for peer-to-peer communication and for accessing the Internet. Such access, including web services or remote storage relying on cloud computing, typically uses SSL/TLS protocol. Often the handling of the key material or the client end of the session needs to be secured.

- Mobile financial services: Some types of financial services tend to be targeted at smart phones, such as mobile banking, mobile money transfer, mobile authentication (e.g. use with One Time Password OTP technology), mobile proximity payments, etc. These services require secure user authentication and secure transaction, which can be performed by the device potentially in cooperation with a Secure Element.

- (Biometric) Authentication services: Such services require a robust root-of-trust, isolation from other execution environments, and the controlled use of the user interfaces, such as biometric sensors and displays.

Refer to the TEE White Paper **[TEEWP]** for an overview of the main TEE use cases.

1.3.3.3 DEVICE LIFE CYCLE

This section exposes the life cycle of the TEE device according to the **[GPD_SPE_021]**, however as the TOE does not contain the hardware elements of the TEE, some of the lifecycle phases are out of scope. Nevertheless, a description of all phases is included in order to fully understand the life cycle.

The device life cycle distinguishes stages for development, OEM integration, device production (device assembling) and operational use (end-usage of the device). The development and production of the TOE together constitute the development phase of the TOE. The development phase is subject of CC evaluation according to the assurance life cycle (ALC) class. The device life cycle is split in six phases:

- Phase 1 corresponds to the design of firmware, software and hardware; it covers both TEE and additional components.

- Phase 2 corresponds to the overall design of the hardware platform supporting the TEE.

- Phase 3 corresponds to chipset and other hardware components manufacturing.

- Phase 4 covers software preparation (e.g. linking the TEE software and other software).

- Phase 5 consists of device assembling; it includes any initialization and configuration step necessary to bring the device to a secure state prior delivery to the end-user.

- Phase 6 stands for the end-usage of the device.

Phases	Actors
1 & 2: Firmware / Software / Hardware design	The TEE firmware/software developer • Is in charge of the development and testing of the Trusted OS components and the GlobalPlatform (and proprietary) APIs. • Does not develop the TEE initialization code that instantiates/initializes the TEE (e.g. part of the secure boot code). It is developed by hardware/firmware team and not included in this document. • Specifies the TEE software linking requirements. The device manufacturer may design additional REE software and Trusted Applications to link with the TEE in phase 4. The TEE hardware designer is in charge of designing (part of) the processor(s) where the TEE runs and the hardware security resources used by the TEE. The silicon vendor designs the ROM code and the secure portion of the TEE chipset. If the silicon vendor is not designing the full TEE hardware, the silicon vendor integrates (and potentially augments) the TEE hardware designed by the TEE hardware designer(s).
3: TEE manufacturing	The silicon vendor produces the TEE chipset and enables, sets or seeds the root of trust of the TEE.
4: Software manufacturing	The device manufacturer is responsible for the integration, validation and preparation of the software to load in the product: the TEE, possibly some pre-installed Trusted Application(s), and additional software required to use the product (e.g. REE, Client Applications).
5: Device manufacturing	The device manufacturer is responsible for device assembly, initialisation, and any other operation on the device

	(including loading or installation of Trusted Applications) before delivery to the end user.
6: End-usage phase	The end user gets a device ready for use. The Trusted Applications manager is responsible for the loading, installation, and removal of Trusted Applications post-issuance.

Table 1 *Actors in the Device Life Cycle*

The phases related to the TOE and in the scope of the evaluation during the whole device life cycle are as below:

- Phase 1, Software design: including TEE software design and development. Delivering the image of TEE software to device manufacturer.

The other phases are out of the scope of this ST.

1.3.4 NON-TOE HARDWARE/SOFTWARE/FIRMWARE

The TOE comprises:

- The software used to provide the TOE security functionality.
- The guidance for the secure usage of the TOE after delivery.

The TOE does not comprise:

- The hardware on which the TOE runs and the firmware running on such hardware.
- The Regular Execution Environment (REE).
- The Trusted Applications (TA), except for the TA that offers the trusted storage service and the Kernel TA.
- The Client Applications (CA).
- ARM Trustzone, which provides isolation in hardware level between the TEE and the REE. It protects the environment by isolating the mechanism in two parts: HLOS and Trusted OS.
- TrustZone Protect Controller, a firewall of Unisoc which provides the functions to configure secure memory and secure peripherals.

- Secure storage key derivation which comes from the HUK key which is implemented in the hardware and is not accessible or retrievable from software.
- The secure loading process.
- Verification of the authenticity of TOS and TEEcfg. The TEEcfg is the TOE configuration, it is not mandatory for the customer to customize TEE functions and they can use the keep the default options.
- Asymmetric atomic cryptographic primitives (for RSA and ECC) that are implemented in Hardware (non-TOE), such as modular operations, random prime generation or padding operations, and symmetric/asymmetric key generation, among others. AES symmetric operations of each round for CBC and ECB modes, which are implemented in Hardware (non-TOE).
- To support third-party TAs, which are always on the REE side, the TOE has functional dependencies with the Tsupplicant to load data from processor memory such as TA code. Tsupplicant is a daemon program located in the REE side that cooperates with TA manager in the TEE for the above purpose.

In the following, TOE and TEE are used interchangeably, except when referring to the device firmware (TEE Firmware) and the device hardware (TEE Hardware).

The Non-TOE hardware/firmware which allows the installation of the TOE on the following Systems on Chip (SoC):

- Mobile series: SC7731E, SC9832E, SC9863A, T310, T610, T618 and T710.
- IoT series UIS8850DG, UIS8850EG.

1.4 TOE DESCRIPTION

1.4.1 TOE LOGICAL SCOPE

The Unisoc Trusted OS implements the TEE software platform on Unisoc SoCs compliant with ARM TrustZone which is a TEE solution designed for the SoC processors based on ARM architecture.

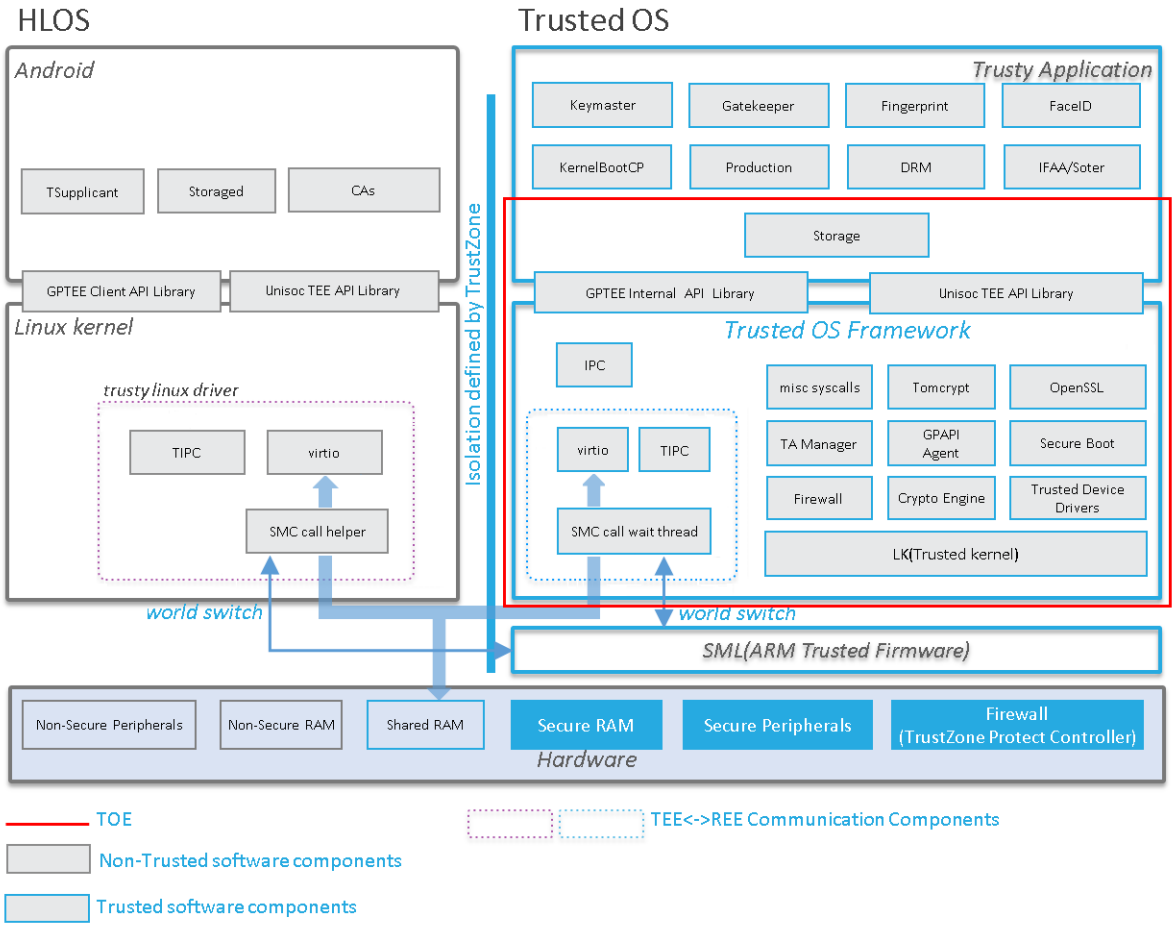


Figure 1 Unisoc Trusted OS Architecture for Android

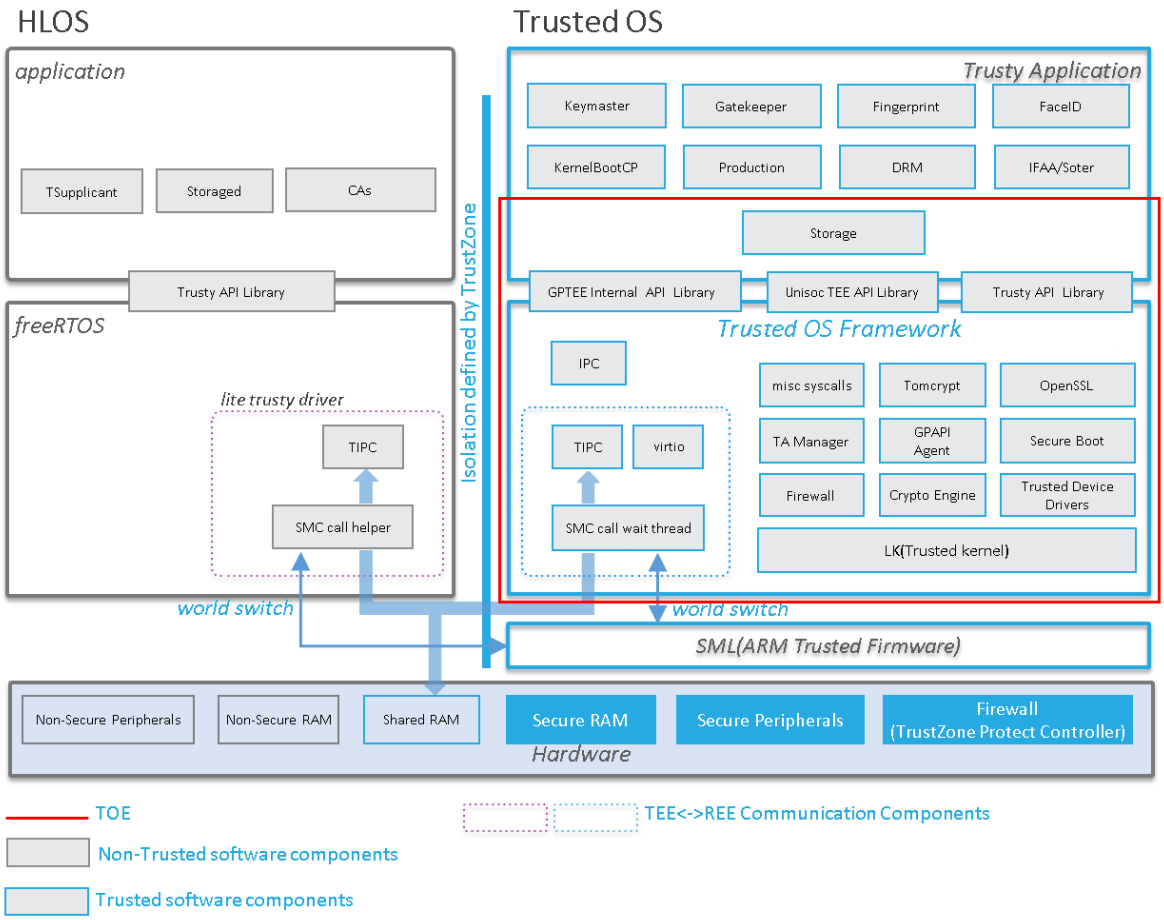


Figure 2 Unisoc Trusted OS Architecture for freeRTOS

The Unisoc TEE OS software (Trusted OS) runs in parallel with HLOS in REE (non-TOE). The REE implements the communication component serving TEE functions to Client Applications (CA) which run on the REE.

The TOS supports two different communication mechanisms that allows different configurations of the REE. On one side, devices running Android OS (Mobile series listed in section 1.3.4) can communicate with the TEE making use of the virtio driver framework (as shown in *Figure 1*). On the other side, devices running Real Time Operating Systems (RTOS) such as *freeRTOS* (IoT devices shown listed in section 1.3.4), communicate with TEE making use of Trusted IPC (TIPC) device driver (as shown in *Figure 2*). The TOE does not need to be adjusted; it offers both mechanisms so the HLOS uses the one that meets the implemented configuration. Both possible configurations are in the scope of the evaluation.

However, the TOE provides a GP TEE APIs in both architectures, although they cannot invoked in freeRTOS architecture (IoT devices shown listed in section 1.3.4). The TOE implements these APIs in both architectures, but these APIs need support of Android in the REE so that they can be exercised.

The TEE OS software architecture identifies two distinct classes of components:

- The Trusted Applications (TA) that run on the TEE and use the TEE Internal API. Trusted Applications are non-TOE components.
- The Trusted OS components whose role is to provide communication facilities with the REE software (non-TOE) and the system level functionality required by the Trusted Applications (TA).

The TOE is composed of the following components:

Unisoc Component	Description
GPTEE Internal API Library	Provides the standard TEE API which is compliance with Global Platform TEE specification. Its implementation is based on Trusty APIs, but provides another approach for the communication between one TA with another TA, or TA and CA. This API is available only when the TOE is deployed in an environment with Android and Linux kernel in the REE, as in *Figure 1*, but it is not accessible when the TOE is deployed in an environment with FreeRTOS in the REE, as in *Figure 2*.
Trusty API Library	Wraps the system calls. It provides the underlying APIs to the TAs for communication with other TAs or Cas, calling OS core functions and accessing hardware devices, etc. This API is only available for freeRTOS architecture, as shown is *Figure 2*.
IPC	TIPC support the communication mechanism between the REE and TEE working with the REE TIPC Driver.
Misc syscalls	Provides all system calls to user space to call various functions of trusted OS. These includes

	IPC communications, time/delay, device control, memory mapping, etc. TAs cannot directly access syscalls, some of the miscellaneous syscall functionality is available through wrappers in the TA APIs.
Virtio	It is a component developed as a standardized open interface used in TEE because Trusty IPC is implemented based on the communication protocol provided by Virtio.
TIPC	It is a specific IPC implementation used for REE/TEE communication. Implements an IPC mechanism for communication between TAs and CAs.
GPAPI Agent	Works as the TEE Communication Agent in the GP standard TEE software architecture.
Secure Boot	The core functions of secure boot are developed in trusted OS, including verification of the TAs code by the TOS, initialisation of the TOS kernel and internal TAs, OTP device programming, which is the important RoT (root of trust) process, etc.
Firewall	It is a driver in Trusted OS which implements the software interfaces to protect RAM, chipset registers and peripheral devices.
Crypto Engine	It is responsible for making calls to cryptographic algorithms, including encryption, decryption, random number generation and key derivation, etc. that are implemented by the hardware.

Trusted Device Drivers	Trusted Device Drivers provide the hardware support for TEE environment.
SMC call wait thread	It is responsible for polling the SMC command executed by REE for REE (TEE switching operation).
LK	Provides the fundamental OS functions, including thread, scheduler, lock, timer, MMU and memory allocator, etc.
Tomcrypt	Is used as the cryptographic library by TAs if TAs are developed based on GPTEE Internal API.
OpenSSL	Is used as the cryptographic library by TAs if TAs are developed based on Trusty API, rather than GPAPI.
Storage	Implements trusted storage service in trusted OS. It also provides the storage operation interfaces to TAs to access trusted storage service, like file create, write, read, etc.
TA Manager	It is responsible for loading the TA firmware from the file system of REE, verifying TA firmware and executing TA in TEE and for reassembling the transmitted TA.
Unisoc TEE API Library	This library is used by TAs communication, CA and TA communication, Trusted Storage and Cryptographic Operations. This API is available for both freeRTOS and Android.

Table 2 *TOE Components*

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

- **Protection of TSF.** The TOE implements a set of mechanisms to ensure its self-security. The main mechanism are as follows:
    - Verification of the authenticity of the TA code before execution
    - TSF failure with preservation of secure state
    - Integrity protection of TEE data
    - Security Management
    - Detect potential security violation
- **Trusted Storage:** is the set of operations to ensure the secure storage of data and keys, including:
    - The confidentiality of the stored data and keys
    - The authenticity of the stored data and keys
    - The consistency of each TA stored data and keys
    - The atomicity of the operations that modify the storage
- **Cryptographic operations:** the algorithms included in the scope of this Security Target are those described in the table below:

Cryptographic Operation	Cryptographic Algorithm	Supported Modes	Key Sizes	Corresponding Standards
Symmetric Cryptography	**AES***	ECB and CBC	128, 192 and 256 bits	**[FIPS 197]** (AES) **[NIST SP800-38A]** (ECB, CBC)
	AES	CTR, CFB, and OFB	128 bits	**[FIPS 197]** (AES) **[NIST SP800-38A]** (CTR, CFB, and OFB)

Cryptographic Operation	Cryptographic Algorithm	Supported Modes	Key Sizes	Corresponding Standards
	DES/TDES	ECB, CBC and CFB	64 bits (DES), 128 or 192 bits (TDES)	**[FIPS 46-3]** (DES, 3DES) **[FIPS 81]** (ECB, CBC, and CFB)
Asymmetric Cryptography	**RSA Encryption***	Padding Scheme NOPAD, RSAES-OAEP-MGF1, RSAES-PKCS1-v1_5	256, 1024, 2048, 3072 and 4096 bits	**[RFC 8017]** (RSA)
	RSA Digital Signature*	Padding Scheme NOPAD, RSASSA_PKCS1_PSS_MGF, RSASSA_PKCS1_v1_5 Hash function SHA224, SHA256, SHA384 and SHA512	256, 1024, 2048, 3072 and 4096 bits	**[RFC 8017]** (RSA) **[FIPS 186-4]** (DSS)
	ECC Digital Signature (ECDSA)*	NIST Curves P-192, P-224, P-256, P-384 and P-521 ANSI X9.63 Curves ansip160k1, ansix9p160r1 and ansix9p160r2	---	**[FIPS 186-4]** (DSS) **[ANSI X9.63]**
	ECC Key Exchange*	NIST Curves P-192, P-224, P-256, P-384 and P-521 ANSI X9.63 Curves ansip160k1, ansix9p160r1 and ansix9p160r2	---	**[RFC 5480]** **[ANSI X9.63]**

* Please refer to FCS_COP.1: Cryptographic operation Application Note because the indicated algorithms are not fully implemented by the TOE because part of them are implemented in Hardware (non-TOE).

Table 3 Supported Cryptographic Algorithms

Key generation is not in the scope of the TOE or the evaluation. The generation of keys is carried out by the non-TOE hardware platform. Therefore, the TOE does not provide any security guarantees for the use of key generation functions.

However, the secure destruction of keys generated by the hardware is included in the TOE scope.

In addition, the TOE provides key import functionality for symmetric and asymmetric cryptographic operations through GPTEE Internal API.

- **User Identification and Authentication:** Both CAs and TAs should be identified and authenticated by the TOE before any more actions.
- **Protection of Trusted Applications (TA):** The TOE provides TA protection during the life cycle from its loading to exiting. For each TA, the TSF will create an isolated user space, and it can't be accessed by other TAs. TAs can access kernel functions only by GP TEE API, and every time it accesses the kernel, the TSF will perform security policy to ensure correct execution.
- **Communication Data Protection between Client Applications (CA) and Trusted Applications (TA):** TAs running in the TOE provide services to CAs running in the REE world, and CAs can communicate with TAs with specified client API. The TOE will protect the whole communication process and communication data.

Note: The security functionality provided by the Trusted Applications is out the scope of the TOE, except for the Trusted Storage TA and the Kernel TA, which are in the scope of the TOE.

1.4.1.1 NON-TOE SECURITY FEATURES

The isolation mechanism between the TEE and the REE mentioned is defined by ARM TrustZone, which provides an isolation environment in hardware level. In this mechanism, all critical CPU resources are separated between Trusted OS and HLOS, like general purpose registers, MMU, data cache and instruction cache, interrupts and etc. The memory space and peripheral devices are separated to secure parts and non-secure parts by the external firewall controller. The former can only be accessed by Trusted OS and the latter can be accessed by both Trusted OS and HLOS, or HLOS only. The only way by which HLOS talks to Trusted OS is the shared RAM provided in HLOS. Generally speaking, the communication channel between TEE and REE is based on shared memory and world switch operation mentioned in the following section.

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

- The Client Applications (CA) which make use of the TEE Client API to access the ensure services offered by TAs running on the TEE.
- The Regular OS, which provides the TEE Client API and sends requests to the TEE.

The task of generating symmetric and asymmetric keys is entrusted to the HW. Given that the TOE exclusively comprises SW, its sole responsibility is to initiate a request to the HW for the generation of the desired key type, which the hardware HW then fulfills.

1.4.2 TOE PHYSICAL SCOPE

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

Title	Version	Format	Distribution Method
Unisoc TEE OS	2.1.2	binary	GIT server
Unisoc TEE OS - AGD_PRE Preparative Guidelines	0.14	PDF	PGP-Encrypted email
Unisoc TEE OS – AGD_OPE Operational guidelines	0.11	PDF	PGP-Encrypted email
Unisoc TEE OS - Functional Specification guidelines	0.15	PDF	PGP-Encrypted email
Unisoc TEE SDK Guide	2.1	PDF	PGP-Encrypted email
UNISOC Research Download User Guide V1.1 EN	1.1	PDF	PGP-Encrypted email
TA Development Guide on Unisoc TEE	0.2	PDF	PGP-Encrypted email
Driver Install and Uninstall Guide (EN)	1.1	PDF	PGP-Encrypted email

Android 11.0 IDH Package Compilation Guide	1.1	PDF	PGP-Encrypted email
GlobalPlatform Technology TEE Internal Core API Specification	1.2.1	PDF	PGP-Encrypted email.

Table 4 Guidance documents

2 CONFORMANCE CLAIMS

This Security Target and the TOE described are in accordance with the requirements of Common Criteria 3.1R5.

This Security Target claims conformance with the following parts of Common Criteria:

- Conformance with [CC31R5P2].
- Conformance with [CC31R5P3] extended.

The methodology to be used for the evaluation is described in the "Common Evaluation Methodology" of the Common Criteria standard of April 2017, version 3.1 revision 5 with an evaluation assurance level of EAL2 + AVA_VAN_AP.3.

This Security Target does not claim conformance with any protection profile.

While the Security Target does not claim conformance to a protection profile, it implements part of the functionality described in the **[GPD_SPE_021]**. The hardware components defined in **[GPD_SPE_021]** are considered as the environment of the TOE, thus the Security Problem Definition (SPD) has been amended, changing hardware-relevant security objectives for the TOE into security objectives for the operational environment and adding new assumptions. Hardware-relevant security functional requirements have been removed from the ST. The two optional PP-Modules TEE Time and Rollback and TEE Debug are not implemented. In addition, attacker profiles have been reduced to SW attackers due to the delegation of all HW functionality to the environment.

The following tables show the relation between the **[GPD_SPE_021]** and this ST in terms of the SPD, the Security Objectives (SO) and the Security Functional Requirements (SFR).

SPDs	**GPD TEE PP**	**Unisoc Included**
Assumptions		

A.PROTECTION_AFTER_DELIVERY	X	X
A.ROLLBACK	X	X
A.TA_DEVELOPMENT	X	X
A.TA_MANAGEMENT	X	X
A.INTEGRATION		X
A.SECURE_HARDWARE_PLATFORM		X
A.SECUREBOOT		X
A.ROOT_KEY		X
Threats		
T.ABUSE_FUNCT	X	X
T.CLONE	X	X
T.FLASH_DUMP	X	X
T.IMPERSONATION	X	X
T.ROGUE_CODE_EXECUTION	X	X
T.PERTURBATION	X	X

T.RAM	X	X
T.RNG	X	X
T.SPY	X	X
T.TEE_FIRMWARE_DOWNGRADE	X	
T.STORAGE_CORRUPTION	X	X
T.ABUSE_DEBUG	X	X
Organizational Security Policies		
OSP.INTEGRATION_CONFIGURATION	X	X
OSP.SECRETS	X	X

Table 5 *Relation of SPDs in the [GPD_SPE_021] and this ST*

Security Objective **Security Objectives for the TOE**	**GPD TEE PP**	**Unisoc Included**
O.CA_TA_IDENTIFICATION	X	X

O.KEYS_USAGE	X	X
O.TEE_ID	X	
O.INITIALIZATION	X	Changed into OE.INITIALIZATION
O.INSTANCE_TIME	X	Changed into OE.INSTANCE_TIME
O.OPERATION	X	X
O.TEE_ISOLATION	X	Changed into OE.TEE_ISOLATION
O.RNG	X	Changed into OE.RNG
O.RUNTIME_CONFIDENTIALITY	X	X
O.RUNTIME_INTEGRITY	X	X
O.TA_AUTHENTICITY	X	X
O.TA_ISOLATION	X	X
O.TEE_DATA_PROTECTION	X	X
O.TRUSTED_STORAGE	X	X

Security Objectives for the Operational Environment		
OE.INTEGRATION_CONFIGURATION	X	X
OE.PROTECTION_AFTER_DELIVERY	X	X
OE.ROLLBACK	X	X
OE.SECRETS	X	X
OE.TA_DEVELOPMENT	X	X
OE.TA_MANAGEMENT	X	X
OE.DISABLED_DEBUG	X	X
OE.TRUSTED_HARDWARE		X
OE.RNG		X
OE.INITIALIZATION		X
OE.INSTANCE_TIME		X
OE.ROOT_KEY		X

Table 6 Relation of SOs in the [GPD_SPE_021] and this ST

2.1.1 OMITTED AND AMENDED OBJECTIVES RATIONALE

- O.TEE_ID: This TEE does not provide means to retrieve an unique identifier.
- O.INITIALIZATION: The initialization process resides in hardware and firmware, both out of the scope of this TOE.
- O.INSTANCE_TIME: The operational environment shall provide instance time.
- O.RNG: The random number generation resides in hardware only.
- OE.TEE_ISOLATION: Isolation between the REE and the TEE is provided by the hardware, via ARM TrustZone.

SFRs in [GPD_SPE_021]	Unisoc Included	Applicable to Hardware/Firmware
1. Identification		
FIA_ATD.1 - User attribute definition	X	
FIA_UID.2 - User identification before any action	X	
FIA_USB.1 - User-subject binding	X	
FMT_SMR.1 - Security roles	X	
2. Confidentiality, Integrity and Isolation		
FDP_IFC.2/Runtime - Complete information flow control	X	

FDP_IFF.1/Runtime - Simple security attributes	X	
FDP_ITT.1/Runtime - Basic internal transfer protection		Hardware Only
FDP_RIP.1/Runtime - Subset residual information protection	X	
FPT_ITT.1/Runtime - Basic internal TSF data transfer protection		Hardware Only
3. Cryptography		
FCS_COP.1 - Cryptographic operation	X	
FDP_ACC.1/TA_Keys - Subset access control	X	
FDP_ACF.1/TA_Keys - Security attribute based access control	X	
FMT_MSA.1/TA_Keys - Management of security attributes	X	
FMT_MSA.3/TA_Keys - Static attribute initialisation	X	
4. Initialization, Operation and Firmware Integrity		

FAU_ARP.1 - Security alarms	X	
FDP_SDI.2 - Stored data integrity monitoring and action	X	
FPT_FLS.1 - Failure with preservation of secure state	X	
FPT_INI.1 - TSF initialisation		Objective for the Operational Environment
FMT_SMF.1 - Specification of Management Functions	X	
FPT_TEE.1 - Testing of external entities	X	
5. TEE Identification		
FAU_SAR.1 - Audit review		Depends on Hardware
FAU_STG.1 - Protected audit trail storage		Depends on Hardware
6. Instance Time		
FPT_STM.1/Instance time - Reliable time stamps		Depends on Hardware
7. Random Number Generator		

FCS_RNG.1 - Random numbers generation		Hardware Only
8. Trusted Storage		
FDP_ACC.1/Trusted Storage - Subset access control	X	
FDP_ACF.1/Trusted Storage - Security attribute based access control	X	
FDP_ROL.1/Trusted Storage - Basic rollback	X	
FMT_MSA.1/Trusted Storage - Management of security attributes	X	
FMT_MSA.3/Trusted Storage - Static attribute initialisation	X	
FDP_ITT.1/Trusted Storage - Basic internal transfer protection	X	
Additional SFRs in this ST		
	FCS_CKM.4 - Cryptographic key destruction	

	FDP_ITC.1: Import of user data without security attributes	

Table 7 Relation of the SFRs in [GPD_SPE_021] and this ST

Given the pre

Application Note

The statement of the security functional requirements relies on the characterization of the TEE in terms of users, subjects, objects, information, user data, TSF data, operations and their security attributes; and the access control and information flow Security Functional Policies (SFP) as defined in Section 7.1.1 of **[GPD_SPE_021]**, except for the following SFRs: **FDP_IFC.2/Runtime** and **FDP_IFF.1/Runtime**. Some of the SFP elements have been modified in this Security Target, the details are in Section 6.1.

3 SECURITY PROBLEM DEFINITION

This section describes the security aspects of the operational environment and its expected use in said environment. It includes the declaration of the TOE operational environment that identifies and describes:

- The alleged known threats that will be countered by the TOE
- The organizational security policies that the TOE has to adhere to
- The TOE usage assumptions in the suggested operational environment.

We will begin defining Assets and Agents of threats.

3.1 ASSETS

RNG: Random Number Generator. #Properties: unpredictable random numbers, sufficient entropy.

Application Note

Random numbers are generated by the TEE hardware (non-TOE), but they become an asset as soon as they are received by the TOE.

TA CODE: The code of the installed Trusted Applications. This data is typically stored in external non-volatile memory which is shared with the REE and potentially accessible by it. #Properties: Authenticity and consistency (which implies runtime integrity).

TA DATA AND KEYS: Data and keys managed and stored by TAs using the TEE security services. Data and keys are owned either by the user (the owner of the TEE-enabled device) or by the TA service provider. This data is typically stored in external non-volatile memory which is shared with the REE and potentially accessible by it. #Properties: Authenticity, consistency (which implies runtime integrity), atomicity and confidentiality.

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

TEE RUNTIME DATA: TEE runtime data includes execution variables, runtime context, etc. This data is stored in volatile memory. #Properties: Consistency (or integrity as these notions are equivalent for non-persistent data) and confidentiality, including random numbers generated by the TEE hardware.

TEE PERSISTENT DATA: TEE persistent data includes cryptographic keys (for instance keys to authenticate TA code) and TA properties. This data is typically stored in external non-volatile memory which is shared with REE and potentially accessible by it. #Properties: Authenticity, consistency (which implies runtime integrity) and confidentiality.

TEE SOFTWARE INITIALISATION CODE AND DATA: Initialisation code and data (for instance cryptographic certificates) used from device power-on up to the complete activation of the TEE security services. Authentication of the TEE is part of its initialization. #Properties: Integrity.

Application Note

Asset TEE STORAGE ROOT OF TRUST is removed in this ST owing to the root of trust is stored in TEE Hardware (non- TOE), and it never reaches inside the TOE.

3.2 THREAT AGENTS

REMOTE ATTACKER: This exploitation profile performs the attack on a remotely-controlled device or makes a downloadable tool that is very convenient to end users. The attacker retrieves details of the vulnerability identified in the identification phase and 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 makes a friendly tool available on the internet. Note that the design of a new malware, Trojan, virus, or rooting tool is often performed from an existing base, available on the internet.

LOCAL LAYMAN ATTACKER: This exploitation profile has physical access to the target device; the end user or someone on his behalf may be the attacker. The attacker may retrieve attack code/application from the identifier and/or guidelines written on the internet on how to perform the attack, and may download and use tools to jailbreak/root/reflash the device in order to get privileged access to the REE allowing the execution of the exploit. These attacks do not require specialized equipment.

Application Note

In the [GPD_SPE_021] four different types of exploitation profiles are defined in section A.2 Attackers' Exploitation Profiles: remote attacker, local layman attacker, local proficient attacker and

local proficient attacker with equipment). In this document the threat agents have been modified, defining only remote attacker, and local layman attacker, to reduce the attacker profiles.

3.3 THREATS TO SECURITY

This section identifies the threats to assets that require protection by the TOE. The threats are defined in terms of assets concerned, attackers and the adverse action that materializes the threat.

T.ABUSE_FUNC: A **Remote attacker** or **Local Layman attacker** accesses TEE functionality outside of their expected availability range, thus violating irreversible phases of the TEE life cycle or state machine.
A **Remote attacker** or **Local Layman 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 software initialisation code and data** (integrity), **TEE runtime data** (confidentiality, integrity), **RNG** (confidentiality, integrity), **TA code** (authenticity, consistency).
Assets threatened indirectly: **TA data and keys** (confidentiality, authenticity, consistency) including instance time.

Application Note

Attack paths may consist in, for instance, using commands in unexpected contexts or with unexpected parameters, impersonating authorized entities, or exploiting badly implemented reset functionality that provides undue privileges.
In particular, a fake application running in the Regular OS which masquerades as a security application running in the TEE can grab PINs and passwords and run the real security application on behalf of the user. However, such a threat cannot be countered by the TEE alone and must be taken into account in the design of the service, for instance by using an applicative authenticated communication channel between the client and the TA.

T.CLONE: A **Remote attacker** or **Local Layman attacker** manages to copy TEE related data from one device to a second device and makes this device accept them as genuine data.
Assets threatened directly: All data and keys (including **TA data and keys**) (authenticity).

T.FLASH_DUMP: A **Remote attacker** or **Local Layman attacker** partially or totally recovers the content of the external Flash in cleartext, thus disclosing sensitive TA or TEE data and potentially allowing the attacker to mount other attacks.
Assets threatened directly (confidentiality, authenticity, consistency): **TA data and keys**, **TEE persistent data**.

Application Note

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

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

T.IMPERSONATION: A **Remote attacker** or **Local Layman 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: A **Remote attacker** or **Local Layman 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 assets.

Application Note

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

T.PERTURBATION: A **Remote attacker** or **Local Layman attacker** modifies the behaviour of the TEE or the behaviour of a TA in order to disclose or modify sensitive data or to force the TEE or the TA to execute unauthorized services.
Assets threatened directly: **TEE software initialisation code and data** (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, for example, exploit multi-threading, context/session management, or closed sessions; or by triggering system resets during execution of commands by the TEE.

T.RAM: A **Local Layman attacker** partially or totally recovers RAM content, thus disclosing runtime data and potentially allowing the attacker to interfere with the **TEE software initialisation code and data**.
Assets threatened directly: **TEE software initialisation code and data** (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 share some memory, an attack path consists in a (partial) memory dump (read/write) by the REE.
During the identification phase, another example of an attack path is to snoop on a memory bus, revealing code that is only decrypted at runtime, and finding a flaw in that code that can be exploited.

T.RNG: A **Remote attacker** or **Local Layman attacker** obtains information in an unauthorized manner about random numbers generated by the TEE. This may occur, for instance, because the generated random numbers have insufficient entropy, or because the attacker forces the output of a partially or totally predefined value.
Loss of unpredictability (the main property of random numbers) is a problem in case they are used to generate cryptographic keys. Malfunctions or premature ageing may also allow getting information about random numbers.
Assets threatened directly (confidentiality, integrity): **RNG** and secrets derived from random numbers.

T.SPY: A **Remote attacker** or **Local Layman attacker** discloses confidential data or keys by means of runtime attacks or by unauthorized access to storage locations.
Assets threatened directly (confidentiality): **TA data and keys**

Application Note

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

T.STORAGE_CORRUPTION: A **Remote attacker** or **Local Layman attacker** corrupts all or part of the non-volatile storage used by the TEE including the trusted storage, in an attempt to trigger unexpected behaviour from the storage security mechanisms. The ultimate goal of the attack is to disclose and/or modify TEE or TA data and/or code.
Assets threatened directly: **TEE persistent data** (confidentiality, consistency), **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.

T.ABUSE_DEBUG: A **Remote attacker** or **Local Layman attacker** manages to be granted access to TEE Debug features, allowing to obtain sensitive data or to compromise the TSF (bypass, deactivate or change security services).
Assets threatened directly: **TEE software initialisation code and data** (integrity), **TEE runtime data** (confidentiality, integrity), **RNG** (confidentiality, integrity), **TA code** (authenticity, consistency).
Assets threatened indirectly: **TA data and keys** (confidentiality, authenticity, consistency) including instance time.

Application Note

During the identification phase, the attacker may search for vulnerabilities, for instance by exploiting the JTAG interface to access the TEE debug mode.

3.4 ORGANIZATIONAL SECURITY POLICIES

The organizational Security policies are defined as follows.

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

Application Note

The integration and configuration of the TEE is by the device manufaturer is defined on the Unisoc TEE Operational Guidelines document.

OSP.SECRETS: Generation, storage, distribution, destruction, or injection of secret data in the TEE and any operation performed on secret data outside the TEE shall enforce the integrity and confidentiality of these data. This applies to secret data injected before the end-usage phase (such as the root of trust of TEE storage) or during the end-usage phase (such as cryptographic private or symmetric keys or any kind of confidential data).

3.5 ASSUMPTIONS

The assumptions when using the TOE are the following:

A.PROTECTION_AFTER_DELIVERY: It is assumed that the TOE and its assets are protected by the operational environment after delivery and before entering the end-usage phase. It is assumed that

the persons using the TOE in the operational environment have the required skills to understand and apply the security guidelines.

Application Note

The protection that is required depends on the delivery point and on the sensitiveness of the assets that are managed between the delivery and the end-usage phase.

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

A.TA_MANAGEMENT: If the TEE allows managing the set of TAs, e.g. updating, replacing, deleting, or installing TAs, then it is assumed that a well-defined TA identification and TA signature policy exist which ensures the authenticity of the application and prevents impersonation. That is, the entity responsible for TA identification and TA signature ensures that these operations are performed in a controlled environment through dedicated procedures, which prevent, by technical and/or organisational means:

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

This applies to all the phases of the TOE life cycle that allow TA management, before and after the TOE delivery point.

Application Note:

The Security Target shall describe the TA management policy.

A.TA_DEVELOPMENT: TA developers are assumed to comply with the guidelines set by the TEE provider. In particular, TA developers are assumed to consider the following principles:

- 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 is not under the TA instance's exclusive control may have changed at next read.
- Reading twice from the same location in memory that is not under the TA instance's exclusive control can return different values.

Application Note:

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

A.ROLLBACK: It is assumed that TA developers do not rely on the protection of TEE persistent data, TA data and keys, and TA code against rollback between two reset operations.

Application Note:

The core TEE PP enforces the consistency (i.e. runtime integrity) of TEE persistent data, TA data and keys, and TA code, but not their integrity between two reset operations.

A.INTEGRATION: It is assumed that the TOE will be installed and configured correctly on the dedicated hardware according to the installation guidance documentation, and all the hardware, firmware components of TEE will be adequately protected. The internal communication data transferred between separated physical components will be protected from disclosure and modification by the TOE environment.

A.SECURE_HARDWARE_PLATFORM: It is assumed that the TOE will run on secure hardware platform defined in **[GPD_SPE_021]**.

- TOE is running in an isolated environment based on ARM Trustzone, which provides protection of internal data transfer between physically separated components of the TOE and isolation between the REE and TEE. In this mechanism, all critical CPU resources are separated between Trusted OS and HLOS, like general purpose registers, MMU, data cache and instruction cache, interrupts and etc.
- For isolation between TAs, the MMU uses separate translation tables for kernel memory space and user memory space to isolate these two spaces. In this way, a TA cannot cross its proprietary memory space to access the memory space of another TA. In this way, isolation between them is achieved.
- It performs secure storage key derivation operation.
- It provides memory protection against tampering and rollback.
- It handles the secure boot process and the authenticity of the TOS and TEEcfg.
- It performs symmetric cryptographic primitives (for AES), such as performing an AES round, and asymmetric ones (for RSA and ECC), such as key pairs generation. Moreover, it implements hashing and MAC operations.
- It implements cryptographic key generation both for symmetric and asymmetric keys.
- It implements sufficient protection over the following type of attacks:
    - Physical attacks
    - Overcoming sensors and filters
    - Perturbation attacks
    - Side-channel attacks

A.SECUREBOOT: It is assumed that the TOE will be started in a secure mode which will ensure:

- The integrity and authenticity of the TEE Firmware.
- The integrity of the TEE initialization code and data.
- The integrity of the storage root of trust (non-TOE).
- The integrity of the TOE software, verified by the TEE firmware (non-TOE).

The version of the firmware to prevent downgrade to previous versions.

The secure boot is performed by the operational environment, by the ROM bootloader (non-TOE) and the SPL (non-TOE).

A.ROOT_KEY: It is assumed that the TEE root of trust (HUK) is stored in the non-TOE Hardware. The key used for trusted storage encryption is derived by the non-TOE Hardware and sent to the TOE.

Application Note

The HUK key itself is never transferred to the TOE according to this assumption. It is kept in the non-TOE Hardware, which is also responsible for key derivation. It is the derived key what is sent to the TOE.

4 SECURITY OBJECTIVES

The security objectives are high level declarations, concise and abstract of the solution to the problem exposed in the former section, which counteracts the threats and fulfills the security policies and the assumptions. These consist of:

- the security objectives for the operational environment.
- the security objectives for the TOE

4.1 SECURITY OBJECTIVES FOR THE TOE

The security objectives for the TOE must determine (to the desired extent) the responsibility of the TOE in countering the threats and in enforcing the OSPs. Each objective must be traced back to aspects of identified threats to be countered by the TOE and to aspects of OSPs to be met by the TOE.

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

Application Note

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

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

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

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

- Control access to its services by the REE and TAs: The TEE shall check the validity of any operation requested from either the REE or a TA, at any entry point into the TEE.
- Enter a secure state upon failure detection, without exposure of any sensitive data.

Application Note

- Programmer errors or violation of good practices (e.g. that exploit multi-threading or context/session management) might become attack-enablers. However, the TEE must guarantee the stability and security of its resources and services independent of the REE, which may have been corrupted. In any case, a Trusted Application must not be able to use a programmer error on purpose to circumvent the TEE security functionality.
- Software in the REE must not be able to call directly to TEE resources or functions. The REE software must go through protocols such that the Trusted OS or Trusted Application performs the verification of the acceptability of the operation that the REE software has requested.

O.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, TEE runtime data, TA code, and TA data and keys are protected against unauthorized modification at runtime when stored in volatile memory.

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

Application Note

Verification of authenticity of TA code is performed 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 integrity of TA data.

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

O.TRUSTED_STORAGE: The TEE shall provide Trusted Storage services for persistent TA data and keys such that the following properties are enforced: confidentiality, authenticity, and consistency. Moreover, the TEE shall either enforce the atomicity of the operations that modify the storage or detect those modifications of the storage have not been completed as expected. The Trusted Storage shall be bound to the host device, which means that the stored data cannot be read in another device.

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

4.2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

The security objectives for the Operational Environment determine the responsibility of the environment in countering the threats, enforcing the OSPs and upholding the assumptions. Each objective must be traced back to aspects of identified threats to be countered by the environment, to aspects of OSPs to be enforced by the environment and to assumptions to be uphold by the environment.

OE.INTEGRATION_CONFIGURATION: Integration and configuration of the TEE by the device manufacturer shall comply with the security guidelines defined by the TEE provider, which must include all recommendations issued from the TOE evaluation. The TOE shall be installed and configured correct to ensure the TOE running in a secure state, including the protection of the internal communication data transferred between separated physical components from disclosure and modification by the TOE environment.

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

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

Application Note

The required protection depends on the delivery point and on the sensitiveness of the assets that are managed between the delivery and the end-usage phase.

OE.TA_MANAGEMENT: If the TEE allows managing the set of TAs, e.g. updating, replacing, deleting, and installing TAs, then a well-defined TA identification and TA signature policy shall ensure the authenticity of the application and prevent impersonation. That is, the entity responsible for TA identification and TA signature shall ensure that these operations are performed in a controlled environment through dedicated procedures, which prevent, by technical and/or organisational means from:

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

Application Note

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

Application Note

TA management is only allowed if the TA authentication performed by signature verification has been successful during the secure initialization.

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

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

- TAs should copy the contents of shared buffers into TA instance-owned memory whenever these contents are required to be constant.

Application Note

The recommendations for the development of secure TAs are defined on the Unisoc TEE Operational Guidelines document.

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

Application Note

The core TEE PP enforces the consistency (i.e. runtime integrity) of TEE persistent data, TA data and keys, and TA code, but not their integrity between two reset operations.

OE.DISABLED_DEBUG: All TEE debug interfaces shall be disabled in the end-user phase.

OE.TRUSTED_HARDWARE: Trusted hardware which the TOE runs on shall provide essential security mechanisms to ensure the TOE runs in a secure state and provide features allowing the TOE to implement secure functionality, upon which the TOE relies on the TEE Hardware (non-TOE). The security functions supported/provided by the trusted platform are:

- A timer to provide reliable timestamps.
- A security mechanism enabling the TOE to isolate different internal processes of the TOE.
- Protection of the memories against tampering.
- A memory that is protected against rollback.
- The secure boot process.
- The TOS and TEEcfg authenticity verification process.
- For isolation between TAs, the MMU is used. It separates translation tables for kernel memory space and user memory space to isolate these two spaces.
- Secure storage key derivation which comes from the HUK key which is implemented in the hardware and is not accessible or retrievable from software.
- Cryptographic Operations: the trusted hardware is responsible for the realization of the symmetric cryptographic primitives of the AES algorithm and the asymmetric cryptographic primitives of the RSA and ECC algorithms. In addition, the trusted hardware fully implements hashing, message authentication code (MAC), and key generation operations.

OE.RNG: The hardware of TEE shall provide a Random Number Generation that is not predictable and that has sufficient entropy.

Application Note

This objective defines the environment functionality to perform random number generation (RNG) operation. This functionality has not been included in the OE.TRUSTED_HARDWARE scope.

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

- The authenticity and integrity of TEE Firmware (non-TOE).
- The integrity of TEE initialization code and data.
- The integrity of the storage root of trust (non-TOE).
- The version of the firmware to prevent downgrade to previous versions.
- The integrity of the TOE software, verified by the TEE firmware (non-TOE).

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 TEE System Architecture (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.ROOT_KEY: The TEE Hardware shall store the root of trust (HUK), and shall derive the key that it is sent to the TOE for trusted storage encryption.

OE.TEE_ISOLATION: The TEE Hardware shall prevent the REE and the TAs from accessing the TEE's own execution and storage space and resources. The security functions supported/provided are:

- A security mechanism enabling the TOE to isolate the REE and the TEE.
- ARM Trustzone for hardware isolation.
- TrustZone Protect Controller, a firewall which provides the functions to configure secure memory and secure peripherals. The memory space and peripheral devices are separated to secure parts and non-secure parts by the external firewall controller.

Application Note

For this TOE, the security mechanism to isolate REE and TEE and different internal processes of the TOE is satisfied by ARM TrustZone technology. This functionality has not been included in the OE.TRUSTED_HARDWARE scope.

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

4.3 SECURITY OBJECTIVES RATIONALE

The following table provides a mapping of security objectives tracing each security objective for the TOE back to threats countered by that security objective and OSPs enforced by that security objective, and each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective. This illustrates that the security objectives counter all threats, the security objectives enforce all OSPs and the security objectives for the operational environment uphold all assumptions.

	O.CA_TA_IDENTIFICATION	O.OPERATION	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TRUSTED_STORAGE	O.KEYS_USAGE	OE.INTEGRATION_CONFIGURATION	OE.SECRETS	OE.PROTECTION_AFTER_DELIVERY	OE.TA_MANAGEMENT	OE.TA_DEVELOPMENT	OE.ROLLBACK	OE.DISABLED_DEBUG	OE.TRUSTED_HARDWARE	OE.RNG	OE.INITIALIZATION	OE.INSTANCE_TIME	OE.ROOT_KEY	OE.TEE_ISOLATION
T.ABUSE_FUNC		X	X	X	X		X		X					X			X		X			X
T.CLONE			X	X			X	X									X		X			X
T.FLASH_DUMP								X									X					
T.IMPERSONATION	X	X		X													X					X
T.ROGUE_CODE_ EXECUTION		X	X	X	X		X	X		X		X					X		X			X
T.PERTURBATION		X	X	X	X	X	X										X		X	X		X
T.RAM			X	X		X											X		X			X
T.RNG			X	X													X	X	X			X
T.SPY			X			X		X									X					X
T.STORAGE_ CORRUPTION		X			X		X	X							X		X		X			

	O.CA_TA_IDENTIFICATION	O.OPERATION	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TRUSTED_STORAGE	O.KEYS_USAGE	OE.INTEGRATION_CONFIGURATION	OE.SECRETS	OE.PROTECTION_AFTER_DELIVERY	OE.TA_MANAGEMENT	OE.TA_DEVELOPMENT	OE.ROLLBACK	OE.DISABLED_DEBUG	OE.TRUSTED_HARDWARE	OE.RNG	OE.INITIALIZATION	OE.INSTANCE_TIME	OE.ROOT_KEY	OE.TEE_ISOLATION
T.ABUSE_DEBUG																X						
OSP.INTEGRATION_ CONFIGURATION										X												
OSP.SECRETS											X											
A.PROTECTION_ AFTER_DELIVERY												X										
A.TA_MANAGEMENT													X									
A.TA_DEVELOPMENT														X								
A.ROLLBACK															X							
A.INTEGRATION										X									X			
A.SECURE_ HARDWARE_ PLATFORM																	X					X
A.SECUREBOOT																			X			

	O.CA_TA_IDENTIFICATION	O.OPERATION	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TRUSTED_STORAGE	O.KEYS_USAGE	OE.INTEGRATION_CONFIGURATION	OE.SECRETS	OE.PROTECTION_AFTER_DELIVERY	OE.TA_MANAGEMENT	OE.TA_DEVELOPMENT	OE.ROLLBACK	OE.DISABLED_DEBUG	OE.TRUSTED_HARDWARE	OE.RNG	OE.INITIALIZATION	OE.INSTANCE_TIME	OE.ROOT_KEY	OE.TEE_ISOLATION
A.ROOT_KEY																					X	

Table 8 Security Objectives vs Security Problem Definition

Figure 3 *Mapping of Security Problem Definition to Security Objectives*

4.3.1 THREATS

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

O.OPERATION ensures correct operation of the security functionality and a proper management of failures.

O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data. **OE.TEE_ISOLATION** provides support to ensure confidentiality by separating the TEE from the outside (REE and TAs).

O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime. **OE.TEE_ISOLATION** provides support to ensure protection against unauthorized modification by separating the TEE from the outside (REE and TAs) and **OE.TRUSTED_HARDWARE** provides support for consistency protection of runtime data.

O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified by the TOE with functional dependence on **OE.TRUSTED_HARDWARE** components for certain cryptographic operations associated with authenticity verification.

O.TEE_DATA_PROTECTION ensures that the data used by the TEE are authentic and consistent. **OE.TRUSTED_HARDWARE** provides a set of cryptographic operations on which the TOE has functional dependencies to perform authenticity and consistency verification.

O.KEYS_USAGE controls the usage of cryptographic keys.

OE.TA_DEVELOPMENT enforces TA development principles, which are meant in particular to prevent disclosing information or performing modifications upon request of unauthorized entities. **OE.INITIALIZATION** ensures that the TEE security functionality is correctly initialized.

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

O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data, particularly TSF data used to bind the TEE to the device. **OE.TEE_ISOLATION** provides support to ensure confidentiality by separating the TEE from the outside (REE and TAs).

O.RUNTIME_INTEGRITY prevents unauthorized modification at runtime of security functionalities or data used to detect or prevent cloning. **OE.TEE_ISOLATION** provides support to ensure protection against unauthorized modification by separating the TEE from the outside (REE and TAs) and **OE.TRUSTED_HARDWARE** provides support for consistency protection of runtime data.

O.TEE_DATA_PROTECTION prevents the TEE from using TEE data that is inconsistent or not authentic. **OE.TRUSTED_HARDWARE** provides a set of cryptographic operations on which the TOE has functional dependencies to perform authenticity and consistency verification.

O.TRUSTED_STORAGE ensures that the trusted storage is bound to the device and prevents the TEE from using data that is inconsistent or not authentic. **OE.TRUSTED_HARDWARE** provides cryptographic support on which the TOE functionally depends to perform portions of the required cryptographic functionality.

OE.INITIALIZATION ensures that the TEE is bound to the SoC of the device.

T.FLASH_DUMP: The combination of the following objectives ensures protection against flash dump attacks:

O.TRUSTED_STORAGE ensures the confidentiality of the data stored in external memory. **OE.TRUSTED_HARDWARE** provides cryptographic support on which the TOE functionally depends to perform portions of the required cryptographic functionality.

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. **OE.TEE_ISOLATION** provides support to ensure protection against unauthorized modification by separating the TEE from the outside (REE and TAs) and **OE.TRUSTED_HARDWARE** provides support for consistency protection of runtime data.

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

O.OPERATION ensures correct operation of the security functionality.

O.RUNTIME_CONFIDENTIALITY covers runtime TEE data which might influence the behaviour of the TEE. **OE.TEE_ISOLATION** provides support to ensure confidentiality by separating the TEE from the outside (REE and TAs).

O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified by the TOE with functional dependence on **OE.TRUSTED_HARDWARE** components for certain cryptographic operations associated with authenticity verification.

O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime. **OE.TEE_ISOLATION** provides support to ensure protection against unauthorized modification by separating the TEE from the outside (REE and TAs) and **OE.TRUSTED_HARDWARE** provides support for consistency protection of runtime data.

O.TEE_DATA_PROTECTION covers persistent TEE data which might influence the behaviour of the TEE. **OE.TRUSTED_HARDWARE** provides a set of cryptographic operations on which the TOE has functional dependencies to perform authenticity and consistency verification.

O.TRUSTED_STORAGE ensures protection of the storage from which code might be imported. **OE.TRUSTED_HARDWARE** provides cryptographic support on which the TOE functionally depends to perform portions of the required cryptographic functionality.

OE.INTEGRATION_CONFIGURATION covers the import of foreign code in a phase other than the end-user phase.
OE.PROTECTION_AFTER_DELIVERY covers the import of foreign code in a phase after delivery and before the end-user phase.
OE.INITIALIZATION ensures that the TEE security functionality is correctly initialized and the integrity of TEE Firmware.

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

O.OPERATION ensures correct operation of the security functionality and proper management of failures.

O.RUNTIME_CONFIDENTIALITY covers runtime TEE data which might influence the behaviour of the TEE. **OE.TEE_ISOLATION** provides support to ensure confidentiality by separating the TEE from the outside (REE and TAs).

O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified by the TOE with functional dependence on **OE.TRUSTED_HARDWARE** components for certain cryptographic operations associated with authenticity verification.

O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime. **OE.TEE_ISOLATION** provides support to ensure protection against unauthorized modification by separating the TEE from the outside (REE and TAs) and **OE.TRUSTED_HARDWARE** provides support for consistency protection of runtime data.

O.TA_ISOLATION ensures the separation of the TA, with the support of **OE.TRUSTED_HARDWARE**.

O.TEE_DATA_PROTECTION covers persistent TEE data which might influence the behaviour of the TEE. **OE.TRUSTED_HARDWARE** provides a set of cryptographic operations on which the TOE has functional dependencies to perform authenticity and consistency verification.

OE.INITIALIZATION ensures that the TEE security functionality is correctly initialized. **OE.INSTANCE_TIME** ensures the reliability of instance time stamps.

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

O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data at runtime. **OE.TEE_ISOLATION** provides support to ensure confidentiality by separating the TEE from the outside (REE and TAs).

O.RUNTIME_INTEGRITY protects against unauthorized modification of code and data at runtime. **OE.TEE_ISOLATION** provides support to ensure protection against unauthorized modification by separating the TEE from the outside (REE and TAs) and **OE.TRUSTED_HARDWARE** provides support for consistency protection of runtime data.

O.TA_ISOLATION provides a memory barrier between TAs, with the support of **OE.TRUSTED_HARDWARE**.

OE.INITIALIZATION ensures that the TEE security functionality is correctly initialized and that the initialization process is protected from the REE.

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

O.RUNTIME_CONFIDENTIALITY ensures that confidential data is not disclosed. **OE.TEE_ISOLATION** provides support to ensure confidentiality by separating the TEE from the outside (REE and TAs).

O.RUNTIME_INTEGRITY protects against unauthorized modification, for instance to force the output of the RNG. **OE.TEE_ISOLATION** provides support to ensure protection against unauthorized

modification by separating the TEE from the outside (REE and TAs) and **OE.TRUSTED_HARDWARE** provides support for consistency protection of runtime data.

OE.INITIALIZATION ensures the correct initialization of the TEE security functions, in particular the RNG.
OE.RNG ensures that random numbers are unpredictable, have sufficient entropy and are not disclosed.

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

O.RUNTIME_CONFIDENTIALITY ensures protection of confidential data at runtime.
OE.TEE_ISOLATION provides support to ensure confidentiality by separating the TEE from the outside (REE and TAs).

O.TA_ISOLATION ensures the separation between TAs, with the support of **OE.TRUSTED_HARDWARE**.

O.TRUSTED_STORAGE ensures that data stored in the trusted storage locations is accessible by the TA owner only. **OE.TRUSTED_HARDWARE** provides cryptographic support on which the TOE functionally depends to perform portions of the required cryptographic functionality.

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.
OE.TRUSTED_HARDWARE provides a set of cryptographic operations on which the TOE has functional dependencies to perform authenticity and consistency verification.

O.TRUSTED_STORAGE enforces detection of corruption of the TA's storage.
OE.TRUSTED_HARDWARE provides cryptographic support on which the TOE functionally depends to perform portions of the required cryptographic functionality.

O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified by the TOE with functional dependence on **OE.TRUSTED_HARDWARE** components for certain cryptographic operations associated with authenticity verification.

OE.INITIALIZATION ensures that the firmware that is executed is the version that was intended.
OE.ROLLBACK states the limits of the properties enforced by the TSF.

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

The following table maps the threats of the security problem established to the security objectives of the TOE and the security objectives of the operational environment.

Threats	Security Objectives
T.ABUSE_FUNC	O.OPERATION O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY O.TA_AUTHENTICITY O.TEE_DATA_PROTECTION O.KEYS_USAGE OE.TEE_ISOLATION OE.TRUSTED_HARDWARE OE.TA_DEVELOPMENT OE.INITIALIZATION
T.CLONE	O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY O.TEE_DATA_PROTECTION O.TRUSTED_STORAGE OE.TEE_ISOLATION OE.TRUSTED_HARDWARE OE.INITIALIZATION

Threats	Security Objectives
T.FLASH_DUMP	O.TRUSTED_STORAGE OE.TRUSTED_HARDWARE
T.IMPERSONATION	O.CA_TA_IDENTIFICATION O.OPERATION O.RUNTIME_INTEGRITY OE.TEE_ISOLATION OE.TRUSTED_HARDWARE
T.ROGUE_CODE_EXECUTION	O.OPERATION O.RUNTIME_CONFIDENTIALITY O.TA_AUTHENTICITY O.RUNTIME_INTEGRITY O.TEE_DATA_PROTECTION O.TRUSTED_STORAGE OE.TEE_ISOLATION OE.TRUSTED_HARDWARE OE.INTEGRATION_CONFIGURATION OE.PROTECTION_AFTER_DELIVERY OE.INITIALIZATION
T.PERTURBATION	O.OPERATION O.RUNTIME_CONFIDENTIALITY O.TA_AUTHENTICITY O.RUNTIME_INTEGRITY O.TA_ISOLATION

Threats	Security Objectives
	O.TEE_DATA_PROTECTION OE.TEE_ISOLATION OE.TRUSTED_HARDWARE OE.INITIALIZATION OE.INSTANCE_TIME
T.RAM	O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY O.TA_ISOLATION OE.TEE_ISOLATION OE.TRUSTED_HARDWARE OE.INITIALIZATION
T.RNG	O.RUNTIME_CONFIDENTIALITY O.RUNTIME_INTEGRITY OE.TEE_ISOLATION OE.TRUSTED_HARDWARE OE.INITIALIZATION OE.RNG
T.SPY	O.RUNTIME_CONFIDENTIALITY O.TA_ISOLATION O.TRUSTED_STORAGE OE.TEE_ISOLATION OE.TRUSTED_HARDWARE
T.STORAGE_CORRUPTION	O.OPERATION

Threats	Security Objectives
	O.TEE_DATA_PROTECTION O.TRUSTED_STORAGE O.TA_AUTHENTICITY OE.TRUSTED_HARDWARE OE.INITIALIZATION OE.ROLLBACK
T.ABUSE_DEBUG	OE.DISABLED_DEBUG

Table 9 Threats vs Security Objectives

4.3.2 ORGANIZATIONAL SECURITY POLICIES

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

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

The following table maps the organizational security policies of the problem established to the security objectives of the TOE and the security objectives of the operational environment.

OSPs	Security Objectives
OSP.INTEGRATION_CONFIGURATION	OE.INTEGRATION_CONFIGURATION
OSP.SECRETS	OE.SECRETS

Table 10 OSPs vs Security Objectives

4.3.3 ASSUMPTIONS

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

A.TA_MANAGEMENT: The objective **OE.TA_MANAGEMENT** directly covers this assumption.

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

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

A.INTEGRATION: The combination of the following objectives covers this assumption:

OE.INTEGRATION_CONFIGURATION ensures that the hardware, firmware and other components will be installed and configured correctly in the non-TOE device.

OE.INITIALIZATION ensures that the firmware installed in the device is the version that was intended.

A.SECURE_HARDWARE_PLATFORM: The combination of the following objectives covers this assumption:

OE.TRUSTED_HARDWARE separates translation tables for kernel memory space and user memory space to isolate these two spaces, for isolation between TAs. In addition, it ensures the secure boot process and the authenticity of the TOS and TEEcfg. Finally, it provides key generation functions and protection against physical attacks.

OE.TEE_ISOLATION ensures the isolation between the REE and TEE.

A.SECUREBOOT: The objective **OE.INITIALIZATION** directly covers this assumption.

A.ROOT_KEY: The objective **OE.ROOT_KEY** directly covers this assumption.

The following table maps the assumptions of the problem established to the security objectives of the TOE and the security objectives of the operational environment.

Assumptions	Security Objectives
A.PROTECTION_AFTER_DELIVERY	OE.PROTECTION_AFTER_DELIVERY
A.TA_MANAGEMENT	OE.TA_MANAGEMENT
A.TA_DEVELOPMENT	OE.TA_DEVELOPMENT
A.ROLLBACK	OE.ROLLBACK
A.INTEGRATION	OE.INTEGRATION_CONFIGURATION OE.INITIALIZATION
A.SECURE_HARDWARE_PLATFORM	OE.TRUSTED_HARDWARE OE.TEE_ISOLATION
A.SECUREBOOT	OE.INITIALIZATION
A.ROOT_KEY	OE.ROOT_KEY

Table 11 *Assumptions vs Security Objectives for the Operational Environment*

5 EXTENDED COMPONENTS DEFINITION

5.1 CLASS AVA: VULNERABILITY ASSESSMENT

A new methodology for evaluating the attack potential for a TEE is needed. Industry stakeholder within the GlobalPlatform TEE Security Working Group and TEE Attacks Expert Working Group have developed such methodology, included in **GPD_SPE_021** Annex A.

5.1.1 VULNERABILITY ANALYSIS OF TEE (AVA_VAN_AP)

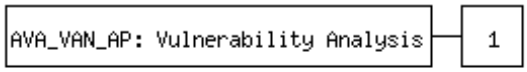
Family objectives

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

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

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

Component levelling



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

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

AVA_VAN_AP.3: TEE Vulnerability Analysis

Hierarchical to:

No other components.

Dependencies:

ADV_ARC.1

ADV_FSP.2

ADV_TDS.1

AGD_OPE.1

AGD_PRE.1

Developer action elements.

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

Content and representation elements.

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

Evaluator action elements.

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

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

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

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

6 SECURITY REQUIREMENTS

This section defines the Security functional requirements (SFRs) and the Security assurance requirements (SARs) that fulfill the TOE. Assignment, selection, iteration and refinement operations have been made, adhering to the following conventions:

- Assignments. They appear between square brackets. The word “assignment” is maintained and the resolution is presented in ***boldface, italic and blue color.***
- Selections. They appear between square brackets. The word “selection” is maintained and the resolution is presented in ***boldface, italic and blue color.***
- Iterations. It includes “/” and an “identifier” following requirement identifier that allows to distinguish the iterations of the requirement. Example: FCS_COP.1/XXX.
- Refinements: the text where the refinement has been done is shown ***bold, italic, and light red colour.*** Where part of the content of a SFR component has been removed, the removed text is shown in ~~***bold, italic, light red colour and crossed out.***~~

The statement of functional security requirements in this section is based on the characterization of the TEE in terms of users, subjects, objects, information, user data, TSF data, operations and their security attributes according to the definitions presented in **[GPD_SPE_021]** (Section 7.1.1), except for the following SFRs: **FDP_IFC.2/Runtime** and **FDP_IFF.1/Runtime**. A number of modifications have been made in these SFRs and in Runtime Data Information Flow Control SFP, removing subjects since the TOE does not perform the task of isolation between the REE and the TEE. This is done by the operational environment.

Modifications are listed below.

Users:

No changes needed.

Subjects:

- *S.RESOURCE* is removed, since it refers to any software or hardware component which may be used alternatively by the TEE, and this separation is now out of scope.

- *S.RAM_UNIT* is modified to consider only the “TA Identifier” security attribute, removing the REE security attribute, since the SFP won’t include REE/TEE separation.

- *S.COMM_AGENT* is kept, but only for the sole purpose of modelling the rules for information flow between the TAs and the communication agent in the TEE, as part of the TA/TEE isolation functionality. It is not considered for isolation between REE and TEE, as that is out of scope.

Information:

No changes needed.

6.1 SECURITY FUNCTIONAL REQUIREMENTS

6.1.1 FAU: SECURITY AUDIT

6.1.1.1 FAU_ARP.1: SECURITY ALARMS

FAU_ARP.1.1 *The TSF shall take* ***[assignment: 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.*

6.1.2 FCS: CRYPTOGRAPHIC SUPPORT

6.1.2.1 FCS_CKM.4: CRYPTOGRAPHIC KEY DESTRUCTION

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

Application Note

The operation of key destruction, as specified in FCS_CKM.4, encompasses:

- Symmetric cryptographic algorithms (AES and DES/TDES) keys, except for the Trusted Storage, where the key is not required to be destroyed. Destruction of these symmetric keys is in the scope of FCS_CKM.4, even when they are not generated by the TOE.
- Asymmetric cryptographic algorithms (ECC and RSA) keys, except for the TA Authentication, where the RSA key is part of the TOE binary and not meant to be destroyed. Destruction of these asymmetric keys is in the scope of FCS_CKM.4, even when they are not generated by the TOE.

6.1.2.2 FCS_COP.1: CRYPTOGRAPHIC OPERATION

FCS_COP.1.1 The TSF shall perform ***[assignment: list of cryptographic operations in Table 3]*** in accordance with a specified cryptographic algorithm ***[assignment: listed in Table 3]*** and cryptographic key sizes ***[assignment: listed in Table 3]*** that meet the following: ***[assignment: corresponding list of standards listed in Table 3].***

Application Note

In some cases, the TOE implements the whole cryptographic operations related to a cryptographic algorithm and, in other cases, the implementation is carried out in a hybrid form by the TOE and the Hardware (non-TOE). In this hybrid form, the TOE is responsible for making an encapsulation for the atomic cryptographic operations that are executed and accelerated by the Hardware (non-TOE).

The cryptographic mechanisms implemented completely by the TOE are as follows:

- AES symmetric cryptography (in modes CTR, CFB, and OFB) with a 128-bit key.

- DES/3DES symmetric cryptography (in modes ECB, CBC, and CFB).

The cryptographic mechanisms implemented in a hybrid form by the TOE and the Hardware are as follows:

- AES symmetric cryptography (in modes ECB and CBC) with 128-bit, 192-bit, and 256-bit keys. For this cryptographic algorithm, the Hardware (non-TOE) implements the atomic symmetric operations at the level of individual blocks by executing the instructions related to each of the AES rounds, and the TOE is responsible for the orchestration of the output of each individual block to constitute the different modes of operation (ECB and CBC).
- RSA asymmetric cryptography. For this cryptographic algorithm, the Hardware (non-TOE) implements the atomic asymmetric operations associated with the processing of each asymmetric instruction (modular exponentiation and other logic operations), and the TOE is responsible for the orchestration of the output of each instruction to constitute the different asymmetric cryptographic operations (asymmetric encryption/decryption and digital signature generation/verification operations).
- ECC asymmetric cryptography. For this cryptographic algorithm, the Hardware (non-TOE) implements the atomic asymmetric operations associated with the processing of each asymmetric instruction (scalar multiplication, curve points processing, and other logic operations), and the TOE is responsible for the orchestration of the output of each instruction to constitute the different asymmetric cryptographic operations (digital signature generation/verification and elliptic curve key exchange operations).

Application Note

TA code is verified using an RSA-2048 signature with SHA-256 as the hash function.

The consistency and confidentiality of Trusted Storage data are protected using a combination of AES-256-CBC, performed in a hybrid manner by Software (TOE) and Hardware (non-TOE), and HMAC SHA-256, performed by Hardware (non-TOE).

The TOE implements other cryptographic operations that can be provided to the TAs as cryptographic services through the API offered to the TAs.

6.1.3 FDP: USER DATA PROTECTION

6.1.3.1 FDP_ACC.1/TRUSTED STORAGE: SUBSET ACCESS CONTROL

FDP_ACC.1.1/TRUSTED STORAGE The TSF shall enforce the ***[assignment: Trusted Storage Access Control SFP]*** on ***[assignment: Subjects: S.API;***
Objects: OB.TA_STORAGE, OB.SRT;
Operations: OP.LOAD, OP.STORE].

6.1.3.2 FDP_ACC.1/TA_KEYS: SUBSET ACCESS CONTROL

FDP_ACC.1.1/TA_KEYS The TSF shall enforce the ***[assignment: TA Keys Access Control SFP]*** on ***[assignment: Subjects: S.API, S.TA_INSTANCE and any other subject in the TEE;***

Objects: OB.TA_KEY;
Operation: OP.USE_KEY, OP.EXTRACT_KEY].

6.1.3.3 FDP_ACF.1/TRUSTED STORAGE: SECURITY ATTRIBUTE BASED ACCESS CONTROL

FDP_ACF.1.1/TRUSTED STORAGE The TSF shall enforce the ***[assignment: Trusted Storage Access Control SFP]*** to objects based on the following: ***[assignment: 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: ***[assignment:***

- ***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).***
    - ***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).***
    - ***If OB.TA_STORAGE is located in external memory accessible to the REE (OB.TA_STORAGE.inExtMem = True) then the object is signed and encrypted before storage.].***

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

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

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

6.1.3.4 FDP_ACF.1/TA_KEYS: SECURITY ATTRIBUTE BASED ACCESS CONTROL

FDP_ACF.1.1/TA_KEYS The TSF shall enforce the ***[assignment: TA Keys Access Control SFP]*** to objects based on the following: ***[assignment: 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: ***[assignment:***

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

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

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

- ***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/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.

6.1.3.5 FDP_IFC.2/RUNTIME: COMPLETE INFORMATION FLOW CONTROL

FDP_IFC.2.1/RUNTIME The TSF shall enforce the ***[assignment: Runtime Data Information Flow Control SFP]*** on ***[assignment: Subjects: S.TA_INSTANCE, S.TA_INSTANCE_SESSION, S.API, 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.

This SFR has been modified from **[GPD_SPE_021]** because the isolation between the TEE and the REE is done by the hardware, via ARM Trustzone (non-TOE). The software is not involved in this isolation process.

6.1.3.6 FDP_IFF.1/RUNTIME: SIMPLE SECURITY ATTRIBUTES

FDP_IFF.1.1/RUNTIME The TSF shall enforce the ***[assignment: Runtime Data Information Flow Control SFP]*** based on the following types of subject and information security attributes: ***[assignment: 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: ***[assignment:***

- ***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.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.***

FDP_IFF.1.3/RUNTIME The TSF shall enforce the ***[assignment: none].***

FDP_IFF.1.4/RUNTIME The TSF shall explicitly authorise an information flow based on the following rules: ***[assignment:***

- ***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: ***[assignment: 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 for TSF data are appropriately protected (in integrity for TEE Firmware, in integrity and confidentiality for TEE runtime data).

This SFR has been modified from **[GPD_SPE_021]** because the isolation between the TEE and the REE is done by the hardware, via ARM Trustzone (non-TOE). The software is not involved in this isolation process.

6.1.3.7 FDP_ITT.1/TRUSTED STORAGE: BASIC INTERNAL TRANSFER PROTECTION

FDP_ITT.1.1/TRUSTED STORAGE The TSF shall enforce the ***[assignment: Trusted Storage Access Control SFP]*** to prevent the ***[selection: disclosure, modification]*** of user data when it is transmitted between physically-separated parts of the TOE.

6.1.3.8 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 ***[selection: deallocation of the resource from]*** the following objects: ***[assignment: 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.

6.1.3.9 FDP_ROL.1/TRUSTED STORAGE: BASIC ROLLBACK

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

FDP_ROL.1.2/TRUSTED STORAGE The TSF shall permit operations to be rolled back within the ***[assignment: failure of any storage operation].***

6.1.3.10 FDP_SDI.2: STORED DATA INTEGRITY MONITORING AND ACTION

FDP_SDI.2.1 *The TSF shall monitor* ~~***user***~~ ***TEE runtime data, TEE persistent data, TA*** *data* ***and keys and TA code*** *stored in containers controlled by the TSF for* ***[assignment: authenticity and consistency errors]*** on all objects, based on the following attributes: ***[assignment: TEE runtime data, TEE persistent data, TA data and keys and TA code].***

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall ***[assignment: behaves in a manner that does not depend on the compromised data; abort the execution of the TA instance; not provide any compromised data.].***

Application Note

The protection of consistency against modification by logical means is done by isolation, for other types of protection hardware is used.

6.1.3.11 FDP_ITC.1: IMPORT OF USER DATA WITHOUT SECURITY ATTRIBUTES

FDP_ITC.1.1 *The TSF shall enforce the* ***[assignment: TAs access keys control FSP]*** *when importing user data, controlled under the SFP, from outside of the TOE.*

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

FDP_ITC.1.3 *The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE:* ***[assignment: verification of the import of all mandatory parameters associated with the imported cryptographic keys, verification of imported symmetric and asymmetric key lengths and consistency of the parameter values, as described in the application note below]****.*

Application Note

The TOE supports the import of symmetric keys (AES and DES/TDES) and asymmetric key pairs (RSA and ECC) via the corresponding GPTEE Internal API Library available for the TAs.

Key import operations can be performed using only GPTEE API, as detailed below:

Cryptographic Operation	Key Type	API	Implementation
Key Import	Symmetric	GPTEE API with Tomcrypt	TOE
Key Import	Asymmetric	GPTEE API with Tomcrypt	TOE
Key Import	Symmetric	Unisoc TEE API	Not Implemented
Key Import	Asymmetric	Unisoc TEE API	Not Implemented

Table 12: Key Import Operation Implementation

For AES symmetric key import, it is verified that the imported key contains the parameter associated with the secret value and has the appropriate length.
For DES/TDES symmetric key import, it is verified that the imported key contains the parameters associated with the secret value and the parity bit buffer, and that it has the appropriate length.
For RSA asymmetric key import, it is verified that the imported key contains the parameters associated with the RSA modulus and the public/private exponent, and that it has the appropriate length. If the associated primes are imported, their consistency is checked by verifying that they are indeed primes.
For ECC asymmetric key import, it is verified that the imported key contains the parameters associated with the ECC curve, the ECC public values (X and Y) and the ECC private value, and has

the appropriate length. The consistency of the parameters is checked to verify that they are indeed associated with the curve.

6.1.4 FIA: IDENTIFICATION AND AUTHENTICATION

6.1.4.1 FIA_ATD.1: USER ATTRIBUTE DEFINITION

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users: ***[assignment: 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 lifetime of this attribute is that of the availability of the TA to clients.
- TA_properties: The lifetime of this attribute is that of the availability of the TA to clients.

6.1.4.2 FIA_UID.2: USER IDENTIFICATION BEFORE ANY ACTION

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

Application Note

User stands for Client Application or Trusted Application.

6.1.4.3 FIA_USB.1: USER-SUBJECT BINDING

FIA_USB.1.1 The TSF shall associate the following user security attributes with subjects acting on the behalf of that user: ***[assignment: 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: ***[assignment: 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: ***[assignment: No modification of client_identity is allowed after initialisation.].***

Application Note

The TOE Internal API defines the codification rules of the CA identity.

6.1.5 FMT: SECURITY MANAGEMENT

6.1.5.1 FMT_MSA.1/TRUSTED STORAGE: MANAGEMENT OF SECURITY ATTRIBUTES

FMT_MSA.1.1/TRUSTED STORAGE The TSF shall enforce the ***[assignment: Trusted Storage Access Control SFP]*** to restrict the ability to ***[selection: query]*** the security attributes ***[assignment: OB.TA_STORAGE.owner, OB.TA_STORAGE.inExtMem, OB.TA_STORAGE.TEE_identity and OB.SRT.TEE_identity]*** to ***[assignment: TA_User role].***

6.1.5.2 FMT_MSA.1/TA_KEYS: MANAGEMENT OF SECURITY ATTRIBUTES

FMT_MSA.1.1/TA_KEYS The TSF shall enforce the ***[assignment: TA Keys Access Control SFP]*** to restrict the ability to ***[selection: change_default, query, modify]*** the security attributes ***[assignment: OB.TA_KEY.usage, OB.TA_KEYS.isExtractable and OB.TA_KEY.owner]*** to ***[assignment: 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].***

6.1.5.3 FMT_MSA.3/TRUSTED STORAGE: STATIC ATTRIBUTE INITIALISATION

FMT_MSA.3.1/TRUSTED STORAGE The TSF shall enforce the ***[assignment: Trusted Storage Access Control SFP]*** to provide ***[selection: restrictive]*** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/TRUSTED STORAGE The TSF shall allow the ***[assignment: TA_User]*** to specify alternative initial values to override the default values when an object or information is created.

6.1.5.4 FMT_MSA.3/TA_KEYS: STATIC ATTRIBUTE INITIALISATION

FMT_MSA.3.1/TA_KEYS The TSF shall enforce the ***[assignment: TA Keys Access Control SFP]*** to provide ***[selection: restrictive]*** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/TA_KEYS The TSF shall allow the ***[assignment: TA_User role]*** to specify alternative initial values to override the default values when an object or information is created.

6.1.5.5 FMT_SMF.1: SPECIFICATION OF MANAGEMENT FUNCTIONS

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

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

6.1.5.6 FMT_SMR.1: SECURITY ROLES

FMT_SMR.1.1 The TSF shall maintain the roles ***[assignment: 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.

6.1.6 FPT: PROTECTION OF THE TSF

6.1.6.1 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: ***[assignment:***

- ***Device binding failure.***
- ***Cryptographic operation failure.***
- ***Invalid CA requests, in particular bad-formed requests.***
- ***Panic states.***
- ***TA code or TA data and 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.

6.1.6.2 FPT_TEE.1: TESTING OF EXTERNAL ENTITIES

FPT_TEE.1.1 The TSF shall run a suite of tests ***[selection: [assignment: prior to execution]]*** to check the fulfillment of ***[assignment: authenticity of TA code].***

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

6.2 SECURITY ASSURANCE REQUIREMENTS

The development and the evaluation of the TOE shall be done in accordance to the following security assurance requirements: **EAL2 + AVA_VAN_AP.3**

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

Assurance Class	Assurance Components
ASE: Security Target evaluation	ASE_CCL.1: Conformance claims ASE_ECD.1: Extended components definition ASE_INT.1: ST introduction ASE_TSS.1: TOE summary specification ASE_OBJ.2: Security objectives ASE_REQ.2: Derived security requirements ASE_SPD.1: Security problem definition
ALC: Life-cycle support	ALC_CMC.2: Use of a CM system ALC_CMS.2: Parts of the TOE CM coverage ALC_DEL.1: Delivery procedures
ADV: Development	ADV_ARC.1: Security architecture description ADV_FSP.2: Security-enforcing functional specification ADV_TDS.1: Basic design
AGD: Guidance documents	AGD_OPE.1: Operational user guidance AGD_PRE.1: Preparative procedures
ATE: Tests	ATE_COV.1: Evidence of coverage ATE_FUN.1: Functional testing ATE_IND.2: Independent testing - sample
AVA: Vulnerability assessment	AVA_VAN.2: Vulnerability analysis AVA_VAN_AP.3: TEE vulnerability analysis

Table 13 *Security Assurance Requirements*

6.2.1 AVA: VULNERABILITY ASSESSMENT

6.2.1.1 AVA_VAN_AP: VULNERABILITY ANALYSIS

6.2.1.1.1 AVA_VAN_AP.3: VULNERABILITY ANALYSIS

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

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

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

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

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

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

6.3 SECURITY REQUIREMENTS RATIONALE

6.3.1 NECESSITY AND SUFFICIENCY ANALYSIS

SFR / TOE Security Objective	O.CA_TA_IDENTIFICATION	O.OPERATION	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TRUSTED_STORAGE	O.KEYS_USAGE
FIA_ATD.1	X	X							
FIA_UID.2	X	X							

SFR / TOE Security Objective	O.CA_TA_IDENTIFICATION	O.OPERATION	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TRUSTED_STORAGE	O.KEYS_USAGE
FIA_USB.1	X	X							
FMT_SMR.1		X							X
FDP_IFC.2/Runtime		X	X	X		X			
FDP_IFF.1/Runtime		X	X	X		X			
FCS_COP.1					X	X	X	X	X
FPT_FLS.1		X				X		X	
FMT_SMF.1		X				X		X	X
FPT_TEE.1					X				
FDP_ACC.1/Trusted Storage		X				X		X	
FDP_ACF.1/Trusted Storage		X				X		X	
FDP_ROL.1/Trusted Storage								X	

SFR / TOE Security Objective	O.CA_TA_IDENTIFICATION	O.OPERATION	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TRUSTED_STORAGE	O.KEYS_USAGE
FMT_MSA.1/Trusted Storage		X				X		X	
FMT_MSA.3/Trusted Storage		X				X		X	
FDP_RIP.1/Runtime			X						
FDP_SDI.2		X		X	X		X	X	
FAU_ARP.1		X							
FDP_ITT.1/Trusted Storage								X	
FDP_ACC.1/TA_keys		X							X
FDP_ACF.1/TA_keys		X							X
FMT_MSA.1/TA_keys		X							X
FMT_MSA.3/TA_keys		X							X
FCS_CKM.4									X

SFR / TOE Security Objective	O.CA_TA_IDENTIFICATION	O.OPERATION	O.RUNTIME_CONFIDENTIALITY	O.RUNTIME_INTEGRITY	O.TA_AUTHENTICITY	O.TA_ISOLATION	O.TEE_DATA_PROTECTION	O.TRUSTED_STORAGE	O.KEYS_USAGE
FDP_ITC.1									X

Table 14 *SFRs / TOE Security Objectives coverage*

6.3.2 SECURITY REQUIREMENT SUFFICIENCY

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

- **FIA_ATD.1** enforces the management of the user identity as security attribute, which then become TSF data, protected in integrity and confidentiality.
- **FIA_UID.2** requires the identification of any user 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.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 behaviour in case of failure.
- **FIA_ATD.1**, **FIA_UID.2** and **FIA_USB.1** ensure that actions are performed by identified users.
- **FMT_SMR.1** states the two operational roles enforced by the TEE.
- **FPT_FLS.1** states that abnormal operations have to lead to a secure state.
- **FDP_ACC.1/Trusted Storage**, **FDP_ACF.1/Trusted Storage**, **FMT_MSA.1/Trusted Storage**, **FMT_MSA.3/Trusted Storage** and **FMT_SMF.1** state the policy for controlling access to TA storage.

- **FDP_IFC.2/Runtime** and **FDP_IFF.1/Runtime** state the policy for controlling access to TA and TEE execution spaces.
- **FDP_ACC.1/TA_keys**, **FDP_ACF.1/TA_keys**, **FMT_MSA.1/TA_keys**, **FMT_MSA.3/TA_keys** and **FMT_SMF.1** state the key access policy.

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

- **FDP_IFC.2/Runtime** and **FDP_IFF.1/Runtime** ensure read access to authorized entities only.
- **FDP_RIP.1/Runtime** states resource clean up policy.

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

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

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.
- **FDP_ACC.1/Trusted Storage**, **FDP_ACF.1/Trusted Storage**, **FDP_ROL.1/Trusted Storage**, **FMT_MSA.1/Trusted Storage**, **FMT_MSA.3/Trusted Storage** and **FMT_SMF.1** state Storage state the policy for accessing TA trusted storage and protecting the confidentiality of data.
- **FDP_SDI.2** enforces the consistency and authenticity of the trusted storage.
- **FDP_ITT.1/Trusted Storage** ensure protection against disclosure of TEE and TA data that is transferred between resources.
- **FPT_FLS.1** maintains a secure state.

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

- **FCS_COP.1** specifies the allowed operations.
- **FCS_CKM.4** specifies key destruction of keys used during symmetric (AES and DES/TDES) and asymmetric operations (RSA and ECC) of Trusted Applications.
- **FDP_ITC.1** specifies key importation that does not have reliable security attributes associated with it.
- **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.

6.3.3 SFR DEPENDENCY RATIONALE

6.3.3.1 TABLE OF SFR DEPENDENCIES

The following table lists the dependencies for each requirement, indicating how they have been satisfied. The abbreviation "h.a." indicates that the dependency has been satisfied by a SFR that is hierarchically above the required dependency.

SFR	Required	Fulfilled	Missing
FIA_ATD.1	None	None	None
FIA_UID.2	None	None	None
FIA_USB.1	FIA_ATD.1	FIA_ATD.1	None
FMT_SMR.1	FIA_UID.1	FIA_UID.2 (h.a. FIA_UID.1)	None
FDP_IFC.2/Runtime	FDP_IFF.1	FDP_IFF.1	None
FDP_IFF.1/Runtime	FDP_IFC.1, FMT_MSA.3	FDP_IFC.2 (h.a. FDP_IFC.1),	FMT_MSA.3
FCS_COP.1	FCS_CKM.4, [FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	See *Table 16* below	FCS_CKM.1
FPT_FLS.1	None	None	None
FMT_SMF.1	None	None	None
FPT_TEE.1	None	None	None
FDP_ACC.1/Trusted Storage	FDP_ACF.1	FDP_ACF.1/Trusted Storage	None
FDP_ACF.1/Trusted Storage	FDP_ACC.1, FMT_MSA.3	FDP_ACC.1/Trusted Storage, FMT_MSA.3/Trusted Storage	None
FDP_ROL.1/Trusted Storage	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1/Trusted Storage	None
FMT_MSA.1/Trusted Storage	FMT_SMR.1, FMT_SMF.1, [FDP_ACC.1 or FDP_IFC.1]	FMT_SMR.1, FMT_SMF.1, FDP_ACC.1/Trusted Storage	None

SFR	Required	Fulfilled	Missing
FMT_MSA.3/Trusted Storage	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1/Trusted Storage, FMT_SMR.1	None
FDP_RIP.1/Runtime	None	None	None
FDP_SDI.2	None	None	None
FAU_ARP.1	FAU_SAA.1	None	FAU_SAA.1
FDP_ITT.1/Trusted Storage	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.1/Trusted Storage	None
FDP_ACC.1/TA_keys	FDP_ACF.1	FDP_ACF.1/TA_keys	None
FDP_ACF.1/TA_keys	FDP_ACC.1, FMT_MSA.3	FDP_ACC.1/TA_keys, FMT_MSA.3/TA_keys	None
FMT_MSA.1/TA_keys	FMT_SMR.1, FMT_SMF.1, [FDP_ACC.1 or FDP_IFC.1]	FMT_SMR.1, FMT_SMF.1, FDP_ACC.1/TA_keys	None
FMT_MSA.3/TA_keys	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1/TA_keys, FMT_SMR.1	None
FCS_CKM.4	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	None	FCS_CKM.1
FDP_ITC.1	[FDP_ACC.1, FDP_IFC.1 or FMT_MSA.3]	FDP_ACC.1/TA_keys, FMT_MSA.3/TA_keys	None

Table 15 *SFR Dependencies*

FCS_COP.1 cryptographic algorithm	Dependency with FCS_CKM.1	Dependency with FDP_ITC.1 or FDP_ITC.2	Dependency with FCS_CKM.4
Hybrid AES Implementation (HW and SW)	Not Applicable The AES keys are generated by Hardware (non-TOE).	Fulfilled	Fulfilled
Software AES Implementation	Not Applicable The AES keys are generated by Hardware (non-TOE).	Fulfilled	Fulfilled
AES-CBC-256 (Trusted Storage)	Not Applicable The AES keys for Trusted Storage are generated by Hardware (non-TOE).	Not Applicable The TOE only keeps the key in volatile memory during the session to perform cryptography. The key is never persisted in non-volatile storage.	Not Applicable The AES keys for Trusted Storage are not erased/destroyed by the TOE.
DES/TDES	Not Applicable The DES keys are generated by Hardware (non-TOE).	Fulfilled	Fulfilled
RSA	Not Applicable The RSA key pairs are generated by Hardware (non-TOE).	Fulfilled	Fulfilled
RSA-2048 (TA Authentication)	Not Applicable The RSA key for TA authentication is part of the TOE binary.	Not Applicable The RSA key for TA authentication is not imported because it is part of the TOE binary.	Not Applicable The RSA key for TA authentication is part of the TOE binary.
ECC	Not Applicable	Fulfilled	Fulfilled

FCS_COP.1 cryptographic algorithm	Dependency with FCS_CKM.1	Dependency with FDP_ITC.1 or FDP_ITC.2	Dependency with FCS_CKM.4
	The ECC key pairs are generated by Hardware (non-TOE).		

Table 16: FCS_COP.1 Dependecies Rationale

Application Note

Key generation is not included as FCS_CKM.1 and, hence, it is not in the scope of the TOE or the evaluation. Therefore, the TOE does not provide any security guarantees for the use of key generation functions. For both symmetric and asymmetric cryptographic algorithms, the entire key generation process is performed by the non-TOE Hardware. The asymmetric RSA key used for TA Authentication is not generated by the TOE, but distributed as part of the TOE binary.

However, destruction of the keys generated by non-TOE is in the scope of the TOE as indicated in FCS_CKM.4.

6.3.3.2 JUSTIFICATION FOR MISSING DEPENDENCIES

FAU_ARP.1 dependency on FAU_SAA.1

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.

FDP_IFF.1/Runtime dependency on FMT_MSA.3

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.

FCS_COP.1 & FCS_CKM.4 dependency on FCS_CKM.1

The task of generating symmetric and asymmetric keys is entrusted to the HW. Given that the TOE exclusively comprises SW, its sole responsibility is to initiate a request to the HW for the generation of the desired key type, which the hardware HW then fulfills.

6.3.4 SAR RATIONALE

The assurance level defined in this Protection Profile consists of the predefined assurance package EAL 2 augmented with AVA_VAN_AP.3 in order to reach the Enhanced-basic attack potential as defined in **[GPD_SPE_021]** (Annex A: “Application of Attack Potential to TEE”).

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

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

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

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

6.3.5 SAR DEPENDENCY RATIONALE

6.3.5.1 TABLE OF SAR DEPENDENCIES

SAR	Required	Fulfilled	Missing
ASE_CCL.1	ASE_INT.1, ASE_ECD.1, ASE_REQ.1	ASE_INT.1, ASE_ECD.1, ASE_REQ.2 (hierarchically above ASE_REQ.1)	None
ASE_ECD.1	None	None	None
ASE_INT.1	None	None	None

SAR	Required	Fulfilled	Missing
ASE_OBJ.2	ASE_SPD.1	ASE_SPD.1	None
ASE_REQ.2	ASE_OBJ.2, ASE_ECD.1	ASE_OBJ.2, ASE_ECD.1	None
ASE_TSS.1	ASE_INT.1, ASE_REQ.1, ADV_FSP.1	ASE_INT.1, ASE_REQ.2 (hierarchically above ASE_REQ.1), ADV_FSP.2 (hierarchically above ADV_FSP.1)	None
ALC_CMC.2	ALC_CMS.1	ALC_CMS.2 (hierarchically above ALC_CMS.1)	None
ALC_CMS.2	None	None	None
ADV_FSP.2	ADV_TDS.1	ADV_TDS.1	None
AGD_OPE.1	ADV_FSP.1	ADV_FSP.2 (hierarchically above ADV_FSP.1)	None
AGD_PRE.1	None	None	None
ATE_IND.2	ADV_FSP.2, AGD_OPE.1, AGD_PRE.1, ATE_COV.1, ATE_FUN.1	ADV_FSP.2, AGD_OPE.1, AGD_PRE.1, ATE_COV.1, ATE_FUN.1	None
AVA_VAN.2	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1	None
ASE_SPD.1	None	None	None
ALC_DEL.1	None	None	None
ADV_ARC.1	ADV_FSP.1, ADV_TDS.1	ADV_FSP.2 (hierarchically above ADV_FSP.1), ADV_TDS.1	None
ADV_TDS.1	ADV_FSP.2	ADV_FSP.2	None
ATE_COV.1	ADV_FSP.2, ATE_FUN.1	ADV_FSP.2, ATE_FUN.1	None

SAR	Required	Fulfilled	Missing
ATE_FUN.1	ATE_COV.1	ATE_COV.1	None
AVA_VAN_AP.3	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1	None

Table 17 SAR dependencies

7 TOE SUMMARY SPECIFICATION

7.1 PROTECTION OF TOE SECURITY FUNCTIONALITY (TSF)

The TOE provides several security mechanisms to ensure TOE's self-protection. Detailed functions include:

- **FPT_TEE.1** ensures that the authenticity of TA code is verified prior to execution.
- **FMT_SMF.1** defines the available management functions and **FMT_SMR.1** maintain the available TSF roles and corresponding associations with users.
- **FDP_SDI.2** monitors stored data, keys, and code, detecting potential integrity violations. In case a consistency or authenticity violation of TA data or code or TA data is detected, **FAU_ARP.1** ensures that the TSF is stopped.
- Upon these or other errors that occur, **FPT_FLS.1** ensures the preservation of a secure state of the TOE. When an error occurs in a cryptographic operation, it may be generated by the Software, if that cryptographic operation is implemented by the TOE, or if it is implemented by the hardware the error may also be detected by the cryptographic hardware. In either case, it will be the TOS that will react to it by carrying out actions to maintain a secure state.

7.2 TRUSTED STORAGE

The TOE provides a secure storage function that performs encryption and tamper-proof data storage services.

The TOE ensures that the necessary Access Control is applied when accessing Trusted Storage (**FDP_ACC.1/Trusted Storage**) based on the corresponding attributes and rules (**FDP_ACF.1/Trusted Storage**) and guarantees rollback of unsuccessful stored information after the failure of the storage operation (**FDP_ROL.1/Trusted Storage**).

The TOE restricts the ability to access stored data (**FMT_MSA.1/Trusted Storage**) based on corresponding security attributes of the stored data that are securely initialized (**FMT_MSA.3/Trusted Storage**).

The algorithm used to encrypt data by file system is the AES-256-CBC (**FCS_COP.1**) and HMAC with SHA-256 (performed by Hardware).

The disclosure or tampering of stored data while transmitting it from Trusted Storage is prevented by **FDP_ITT.1/Trusted Storage**.

7.3 CRYPTOGRAPHIC OPERATIONS

The TOE provides cryptographic operations, using different software libraries depending on the API used (**FCS_COP.1**).

In case the TOE fully implements the cryptographic algorithm, Tomcrypt implements all the cryptographic operations of these algorithms. In case the cryptographic algorithm is implemented in a hybrid form, rather Tomcrypt or OpenSSL are in charge of making requests to the Hardware (non-TOE) to perform the atomic cryptographic instructions and to orchestrate the outputs provided to constitute the different cryptographic operations.

AES symmetric operations are performed by the TOE and the Hardware (non-TOE), only for ECB and CBC. For the CTR, OFB, and CFB modes, the TOE is in charge of the entire cryptographic implementation.

DES/TDES symmetric operations in ECB, CBC, and CFB modes are fully implemented by the TOE.

RSA/ECC asymmetric operations are performed by the TOE and the Hardware (non-TOE). The TOE is in charge of orchestrating the atomic primitives that are implemented and accelerated in the Hardware (non-TOE).

The TOE provides symmetric key (AES and DES/TDES) destruction functionalities (**FCS_CKM.4**). Symmetric key generation functionality is performed by Hardware (non-TOE) and it is out of the scope of the TOE and the evaluation.

For asymmetric operations (RSA and ECC), the TOE only provides key pair destruction functionality (**FCS_CKM.4**). Asymmetric key generation functionality is performed by Hardware (non-TOE) and it is out of the scope of the TOE and the evaluation.

The TOE provides key import functionality (**FDP_ITC.1**) for symmetric and asymmetric cryptographic operations.

The API provides functions for the setting of symmetric keys (AES and DES/TDES) but the key import functionality is not considered because the calling TA is responsible for the management of the storage memory of these keys and the TOE only handles the pointer to it for symmetric cryptographic operations. The same applies to the setting of asymmetric keys for encryption or signature operations, where the TOE only handles the pointer to these keys and the calling TA is responsible for the key memory management.

The TOE ensures that the necessary Access Control is applied when accessing TA keys (**FDP_ACC.1/TA_keys**) based on the corresponding attributes and rules (**FDP_ACF.1/TA_keys**).

The TOE restricts the ability to access TA keys (**FMT_MSA.1/TA_keys**) based on the corresponding security attributes of the keys, that are securely initialized (**FMT_MSA.3/TA_keys**).

7.4 USER IDENTIFICATION AND AUTHENTICATION

The TOE defines the necessary security attributes of Trusted and Client Applications (**FIA_ATD.1**) and associates such attributes with the corresponding user (**FIA_USB.1**).
The TOE requires them to be identified before allowing any TEE operation (**FIA_UID.2**)

7.5 COMMUNICATION DATA PROTECTION BETWEEN CA AND TA

The communication between CAs and TAs is supported by the underlying REE and TEE communication mechanism that needs to solve security and synchronization of data access.. For each loaded TA, the TSF will create an isolated running environment, and any reading or writing accesses from other TAs are forbidden. (**FDP_IFC.2/Runtime** and **FDP_IFF.1/Runtime**).

When a TA requests a new memory area, before passing it to the TA, the kernel overwrites this fragment of memory with a pattern. This way all residual information is erased (**FDP_RIP.1/Runtime**).

8 ACRONYMS

The following table shows the acronyms used in this document.

Acronym	Meaning
PP	Protection Profile
CC	Common Criteria
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFi	TSF Interface
OSP	Organisational Security Policies
EAL	Evaluation Assurance Level
ST	Security Target
IT	Information Technology
TEE	Trusted Execution Environment
TOS	Trusted Operating System
REE	Regular Execution Environment
TA	Trusted Application
CA	Client Application
VPN	Virtual Private Network
IT	Information Technology
HD	High-definition
DRM	Digital Rights Management
SSL	Secure Sockets Layer
TLS	Transport Layer Security
IPsec	Internet Protocol security
OTP	One Time Programmable
ROM	Read Only Memory
RAM	Random Access Memory
OS	Operating System
SoC	System-on-Chip
API	Application Programming Interface
AES	Advanced Encryption Standard
RFC	Request For Comments; may denote a memorandum published by the IETF
DES	Data Encryption Standard
RSA	Rivest / Shamir / Adleman asymmetric algorithm

Acronym	Meaning
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman
SPD	Security Problem Definition
SO	Security Objective
SFR	Security Functional Requirement
USB	Universal Serial Bus
OSP	Organisational Security Policies
EAL	Evaluation Assurance Level
SML	Secure Monitor Layer
HUK	Hardware Unique Key
HLOS	High-Level Operating System
CPU	Central Processing Unit
TIPC	Trusty IPC (Inter-Process Communication)
SPL	Secondary Program Loader

Table 18 *Abbreviations*

9 GLOSSARY OF TERMS

Term	Meaning
Augmentation	Addition of one or more requirement(s) to a package
Evaluation Assurance Level	Set of assurance requirements drawn from CC Part 3, representing a point on the CC predefined assurance scale, that form an assurance package
Operational Environment	Environment in which the TOE is operated
Protection Profile	Implementation-independent statement of security needs for a TOE type
Security Target	Implementation-dependent statement of security needs for a specific identified TOE
Target Of Evaluation	Set of software, firmware and/or hardware possibly accompanied by guidance
Application Programming Interface (API)	A set of rules that software programs can follow to communicate with each other.
Client Application (CA)	An application running outside of the Trusted Execution Environment (TEE) making use of the TEE Client API that accesses facilities provided by the Trusted Applications inside the TEE. Contrast Trusted Application.
Consistency	A property of the TEE persistent storage that stands at the same time for runtime and startup consistency.
Device binding	The property of data being usable only on a unique given system instance, here a TEE.
Execution Environment (EE)	An environment that hosts and executes software. This could be an REE, with hardware hosting Android, Linux, Windows, an RTOS, or other software; it could be a Secure Element or a TEE.
Integrity	Property of the TEE persistent storage that means that the value successfully read from a storage location is the last value that was written to this location.
Monotonicity	The property of a variable whose value is either always increasing or always decreasing over time.
Power cycle	The lapse between the moment a device is turned on and the moment the device is turned off afterwards.
Production TEE	A TEE residing in a device that is in the end-user phase of its life cycle.
REE Communication Agent	REE Regular OS driver that enables communication between the REE and the TEE. Contrast TEE Communication Agent.
Regular Execution Environment (REE)	An Execution Environment comprising at least one Regular OS and all other components of the device (SoCs, other discrete components, firmware, and software) which execute, host, and support the Regular OS (excluding any Secure Components included in the device). From the viewpoint of a Secure Component, everything in the REE is considered untrusted, though from the Regular OS point of view there may be internal trust structures.

Term	Meaning
	(Formerly referred to as a Regular Execution Environment (REE).) Contrast Trusted Execution Environment.
Regular OS	An OS executing in a Regular Execution Environment. May be anything from a large OS such as Linux down to a minimal set of statically linked libraries providing services such as a TCP/IP stack. (Formerly referred to as a Regular OS or Device OS.) Contrast Trusted OS.
Root of Trust (RoT)	Generally the smallest distinguishable set of hardware, firmware, and/or software that must be inherently trusted and which is closely tied to the logic and environment on which it performs its trusted actions.
Secure Component	GlobalPlatform terminology to represent either a Secure Element or a Trusted Execution Environment.
Secure Element	A tamper-resistant secure hardware component which is used in a device to provide the security, confidentiality, and multiple application environment required to support various business models. May exist in any form factor, such as embedded or integrated SE, SIM/UICC, smart card, smart microSD, etc.
System-on-Chip (SoC)	An electronic system all of whose components are included in a single integrated circuit.
TA instance time / TA persistent time	Time value available to a Trusted Application through the TEE Internal API. The API offers two types of time values: System Time, which exists only during runtime, and Persistent time, which persists over resets. System Time must be monotonic for a given TA instance, and the returned value is called “TA instance time”. Persistent time depends only on the TA but not on a particular instance, it must be monotonic even across power cycles. Its monotonicity across power cycles is related to the Time and Rollback optional PP-module.
TEE Client API	The software interface used by clients running in the REE to communicate with the TEE and with the Trusted Applications executed by the TEE.
TEE Communication Agent	TEE Trusted OS driver that enables communication between REE and TEE. Contrast REE Communication Agent.
TEE Internal Core API	The software interface exposing TEE functionality to Trusted Applications.
TEE Service Library	A software library that includes all security related drivers.
Trusted Application (TA)	An application running inside the Trusted Execution Environment that provides security related functionality to Client Applications outside of the TEE or to other Trusted Applications inside the TEE. Contrast Client Application.
Trusted Applications Manager	Entity responsible for loading, installation, and removal of Trusted Applications.
Trusted Execution Environment (TEE)	An Execution Environment that runs alongside but isolated from an REE. A TEE has security capabilities and meets certain security-related requirements: It protects TEE assets against a set of defined threats which include general software attacks as well as some hardware attacks, and defines rigid safeguards as to data and functions that a program can access.

Term	Meaning
	There are multiple technologies that can be used to implement a TEE. Contrast Regular Execution Environment.
Trusted OS	An OS executing in a Secure Component. Contrast Regular OS.
Trusted Storage	In GlobalPlatform TEE documents, trusted storage indicates storage that is protected to at least the robustness level defined for OMTP Secure Storage (in section 5 of [OMTP-TR1]). It is protected either by the hardware of the TEE, or cryptographically by keys held in the TEE. If keys are used they are at least of the strength used to instantiate the TEE. A GlobalPlatform TEE Trusted Storage is not considered hardware tamper resistant to the levels achieved by Secure Elements.

Table 19 *Glossary of terms*

10 DOCUMENT REFERENCES

The following table shows the acronyms used in this document.

Reference	Document
[CC31R5P1]	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, Part 1: Introduction and general model.
[CC31R5P2]	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, Part 2: Security functional components.
[CC31R5P3]	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, Part 3: Security assurance components.
[CEM31R5P3]	Common Criteria Evaluation methodology, Version 3.1, Revision 5.
[GPD_SPE_021]	GlobalPlatform Technology TEE Protection Profile, version 1.3, July 2020
[TEEWP]	The Trusted Execution Environment: Delivering Enhanced Security at a Lower Cost to the Mobile Market, GlobalPlatform White paper, June 2015
[FIPS 186-4]	National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal Information Processing Standards Publication FIPS PUB 186-4, July 2013
[GPD_SPE_010]	GlobalPlatform Technology TEE Internal Core API Specification v1.2.1, May 2019
[NIST SP800-38A]	Recommendation for Block Cipher Modes of Operation. Methods and Techniques. NIST Special Publication 800-38A, 2011 Edition
[FIPS 197]	Federal Information Processing Standards Publication 197 November 26, 2001 Announcing the ADVANCED ENCRYPTION STANDARD (AES)
[FIPS 180-4]	FIPS PUB 180-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Secure Hash Standard (SHS)
[RFC-4231]	Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512

Reference	Document
	(https://datatracker.ietf.org/doc/html/rfc4231)
[RFC 8017]	PKCS #1: RSA Cryptography Specifications Version 2.2 (https://datatracker.ietf.org/doc/html/rfc8017)
[FIPS 46-3]	DATA ENCRYPTION STANDARD (DES), National Institute of Standards and Technology, May 2005
[FIPS 81]	DES Modes of Operation, National Institute of Standards and Technology, May 2005
[NIST SP 800-38B]	Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication, National Institute of Standards and Technology, May 2005
[NIST SP800-38E]	Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices, National Institute of Standards and Technology, January 2010
[RFC-3610]	Counter with CBC-MAC (CCM) (https://datatracker.ietf.org/doc/html/rfc3610)
[ANSI X9.63]	Public Key Cryptography for the Financial Services Industry Key Agreement and Key Transport Using Elliptic Curve Cryptography. ANSI, November 2001
[RFC 5480]	Elliptic Curve Cryptography Subject Public Key Information (https://datatracker.ietf.org/doc/html/rfc5480)
[NIST SP800-38A Addendum]	Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode, October 2010
[TA Development Guide on Unisoc TEE]	Unisoc provides guidance on dynamic TA developing, installing and debugging v0.2
[Unisoc TEE SDK Guide]	Unisoc guide about TEE SDK usage and GPTEE TA development v2.1
[Driver Install and Uninstall Guide (EN)]	Unisoc guide about installation and uninstallation of Drivers v1.1
[UNISOC Research Download User Guide V1.1 EN]	Unisoc guide about flashing TEE in Hardware v1.1

Reference	Document
[100956_Android11.0IDHPackageCompilationGuideV1.1]	Unisoc guide about prepare build environment, build images and generate pac file v1.1

Table 20 *List of document references*