Huawei iTrustee V3.0 on Kirin 980 Security Target

Issue 1.9

Date 2019-11-07

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions

and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: http://www.huawei.com

Email: support@huawei.com

About This Document

Purpose

This document is the Security Target.

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Date	Revision Version	Section Number	Change Description	Author
2019.01.18	1.0	ALL		Zhang Jing Yang Ze Tan Jingguo Yao Dongdong Boutheina Chetali
2019.04.10	1.4	ALL	Draft for registration	
2019.06.14	1.8	ALL	Updates for RE0	All
2019.11.07	1.9	1.2	Ref of TOE	

Contents

Figures

Tables

1 ST Introduction

1.1 ST Reference

Title : Huawei iTrustee V3.0 on Kirin980 Security Target

Version: 1.9

Reference : Huawei iTrustee V3.0 on Kirin980 Security Target V1.9

Author: Huawei Technologies Co., Ltd.

Date of publication: 2919-11-07

ITSEF : Thales

1.2 TOE Reference

Product Name : Huawei iTrustee V3.0 on Kirin980

TOE Components		Developer
SoC reference	Kirin SoC 980	Hisilicon
Boot code ref	Fastboot Version 1.0 - MD5 Hash: 2561c3f407199abcb18633155df398ad	Hisilicon
ATF binary ref	ATF Version 1.0 – MD5 Hash: b105cc4d51511f4b4290f48e2c782aab	ARM/ Hisilicon
TEE binary ref:	iTrustee Version 3.0 – MD5 Hash: 79d5fd6ea04d43134e12dd94218fc612	Huawei Technologies Co., Ltd

1.3 TOE Overview

1.3.1 TOE Type

The TOE type is a Trusted Execution Environment (TEE) for embedded devices implementing GlobalPlatform TEE specifications (see TEE System Architecture [SA], TEE Internal API [IAPI] and TEE Client API [CAPI]).

The TOE is an execution environment isolated from any other execution environment, including the usual Rich Execution Environment (REE), and their applications. The TOE hosts a set of Trusted Applications (TA) and provides them with a comprehensive set of security services including: integrity of execution, secure communication with the Client Applications (CA) running in the REE, trusted storage, key management and cryptographic algorithms, time management and arithmetical API.

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

We refer to the TEE White Paper [WP] for an overview of the main TEE use cases.

1.3.3 TOE and Non-TOE parts

The TOE comprises:

- The hardware, firmware and software used to provide the TEE security functionalities:
    - Hardware part components of Kirin 980 SoC running in the secure world (including RAM, Crypto Accelerators, Processing Cores, ROM, Timer, OTP Fields).
    - Firmware: Security Boot Firmware, Arm Trusted Firmware (ATF)

- Software: iTrustee OS

- The preparative guidance for the secure usage of the TOE after delivery.

Table 1-1 : TOE components

Type	Name	Version
Hardware	Kirin Soc Chip	980
Firmware	FastBoot	1.0
	ATF	1.0
Software	iTrustee	3.0
Documents	Preparative Procedures for User	Huawei iTrustee V3.0 on Kirin 980 AGD_PRE_V1.2
	Operational User Guidance	Huawei iTrustee V3.0 on Kirin 980 AGD_OPE_V1.6

The TOE does not comprise:

- The Trusted Applications
- The Rich Execution Environment
- The Client Applications.

The TOE is an important component of Huawei Mate/P series Mobile Phones, and the necessary Non-TOE hardware in the mobile phone consists at least the following parts:

- Flash memory, where Android OS and iTrustee stored
- DDR memory.
- CPU cores not running in the secure world, GPU.
- Necessary peripherals such as display screen, camera, microphone, power button etc.

The following Non-TOE software is also required for the mobile phone:

- EMUI, Huawei customized Android OS
- Device driver for iTrustee in Android
- SDK for Android app to communicate with iTrustee.

1.4 TOE Description

1.4.1 TOE Physical Architecture

The hardware architecture of the TOE includes: RAM; Crypto Accelerators; Processing Cores; Timer; ROM and OTP, as shown in the figure below:

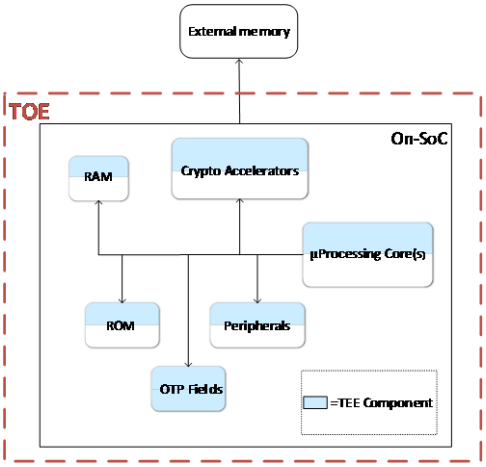

Figure 1-1 The hardware architecture overview of the TOE

1.4.2 TOE Logical Architecture

The TOE is embedded in the device and runs alongside a standard OS or the Rich Execution Environment. Figure 1-2 provides a high level view of the software and firmware components of a TEE-enabled device.

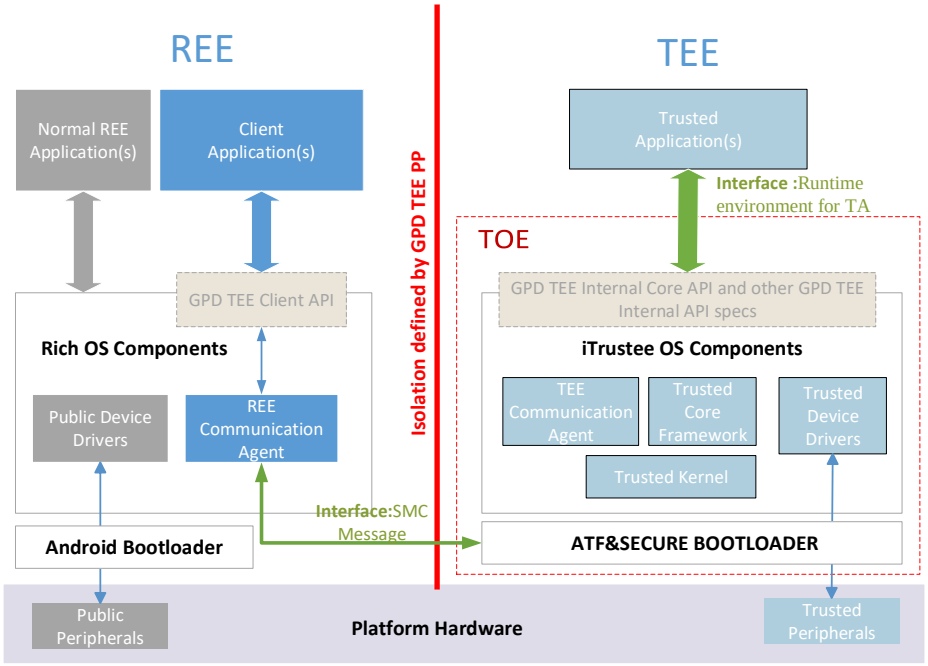

Figure 1-2 TOE software & firmware architecture overview

The TOE software architecture identifies two distinct classes of components:

- The Trusted Applications that run on the TEE and use the TEE Internal API
- The Trusted OS Components whose role is to provide communication facilities with the REE software and the system level functionality required by the Trusted Applications, accessible from the TEE Internal API

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

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

The TOE software external interface comprises the TEE Internal API (used by the Trusted Applications) and the TEE Communication Agent protocol (used by the REE).

The TOE implements self-defined SMC Calls for communication between the REE and the TEE used below the TEE Client API level.

The trusted peripherals include timer modules which are provided by the SoC. The TOE acquires reliable time and RNG from the trusted peripherals via the Trustzone internal Hardware interface.

The TOE software is composed of four components as below:

Table 1-2 iTrustee software components

iTrustee Component	Description
iTrustee Kernel	iTrustee Kernel provides task creation, memory management, IPC functions etc., and provides privileged interface to Trusted Core Framework for system management.
TEE Communication Agent	TEE communication Agent will send/receive SMC calls to/from REE, resolve SMC messages and forward messages to other components of the TOE.
Trusted Core Framework	Trusted Core Framework performs real system management such as TA loading and management and internal communication between TAs. Trusted Core Framework also provides secure storage and log management for TAs.
Trusted Device Drivers	Trusted Device Drivers will talk to Trusted Peripherals, and provide trusted functions to the TOE.

The TOE firmware includes Secure Boot firmware and ATF firmware.

- The Secure Boot firmware make sure that the TOE boots using only software that is signed by the manufacturer by verifying the chain of trust. Any modification of software will be detected and the boot process will stop.
- The ATF firmware manages to transfer SMC message between REE and TEE in a safe way.

1.4.3 TOE Security Functionality

The TOE security functionalities in the end-user phase and in the scope of the evaluation are:

- **Protection of TSF**: Including TSF secure initialization, TSF failure with preservation of secure state, isolation between REE and TEE, integrity protection of TEE data, and security management.
- **Trusted Storage**: Providing confidentiality, consistency and integrity protection of TA data; Binding TA data to the TEE.
- **Reliable time stamps:** providing reliable, monotonic secure clock whenever TOE falling into deep sleep or not, and it will be reset if the TOE is restarted.
- **User Identification and Authentication:** Both CAs and TAs should be identified and authenticated by the TOE before any more actions.
- **Security Audit:** The TOE detect potential security violation such as violation of TA data, TA code or TEE data, and generate audit log. The log will be sent to REE.
- **Protection of TA:** The TOE provide 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 internal core API, and every time it access the kernel, the TSF will perform security policy to ensure correct execution.
- **Communication Data Protection between CA and TA**. TAs running in the TOE provide services to CAs running in the REE world, and CAs can communicate to TAs with specified client API. The TOE will protected the whole communication process and communication data.
- **Security Instantiation** TEE instantiation through a secure initialization process using assets bound to the SoC, that ensures the authenticity and contributes to the integrity of the TEE code running in the device.
- **Cryptographic Support:** Providing cryptographic service to TA via TEE Internal API and Security boot of TOE.

Table 1-3 Cryptographic algorithms supported by the TOE

Algorithm name	Supported Modes	Key sizes (default bits)	Supported standard
Symmetric ciphers(AES)	CBC CTS XTS CTR OFB	128, 192, 256, 512 (XTS)	FIPS 197 (AES) NIST SP800-38A (CBC, CTR) IEEE Std 1619-2007 (XTS) NIST SP800-38A Addendum (CTS = CBC-CS3)
Symmetric ciphers (3DES)	CBC	128 or 192	FIPS 46 (DES, 3DES) FIPS 81 (CBC)
Hashing	SHA1 SHA224 SHA256 SHA384 SHA512		FIPS 180-4 (SHA1 SHA224 SHA256

Algorithm name	Supported Modes		Key sizes (default bits)	Supported standard
				SHA384 SHA512)
MAC(HMAC)	SHA1 SHA224 SHA256 SHA384 SHA512		**block size not exceed**（64bytes for SHA-1 and SHA-224/256, 128bytes for SHA-384/512）	RFC 2202 (SHA1) RFC 4231 (SHA224 SHA256 SHA384 SHA512)
MAC(AES_MAC)	AES_MAC CMAC GCM GMAC		128，256	NIST SP800-38B
Authen EncryModes	AES_CCM		128, 192, 256	RFC 3610 (CCM)
Asymmetric cipher	encrypt/decrypt	TEE_ALG_RSA_NOPAD TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512 TEE_ALG_RSAES_PKCS1_V1_5	2048,3072	PKCS #1 (RSA, PKCS1 v1.5, OAEP) FIPS 180-4 (SHA-1, SHA-2)
	sign/verify	TEE_ALG_RSASSA_PKCS1_V1_5_SHA224 TEE_ALG_RSASSA_PKCS1_V1_5_SHA256 TEE_ALG_RSASSA_PKCS1_V1_5_SHA384 TEE_ALG_RSASSA_PKCS1_V1_5_SHA512 TEE_ALG_RSASSA_PKCS1_PSS_	2048,3072	PKCS #1 (RSA, PKCS1 v1.5, PSS) FIPS 180-4 (SHA-2)

Algorithm name	Supported Modes		Key sizes (default bits)	Supported standard
		MGF1_SHA224 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512		
Key Derivation	DH		2048	PKCS #3 FIPS 186-4* ANSI X9.62 NIST SP800-56A, Cofactor Static Unified Model FIPS 186-4* (curve definitions)
ECC	ECDSA ECDH		160, 192, 224, 256, 384 and 521	FIPS 186-4* ANSI X9.62

The TOE security functionalities define the logical boundary of the TOE. The interfaces of this boundary are the Hardware and Software External Interface, introduced in section 1.4.1 and 1.4.2 respectively.

The security functionality provided by the Trusted Applications is out of the scope of the TOE.

1.5 TOE's Device Life Cycle

The device life cycle outlined here is an overall life cycle from which implementations can deviate according to development, manufacturing and assembly processes.

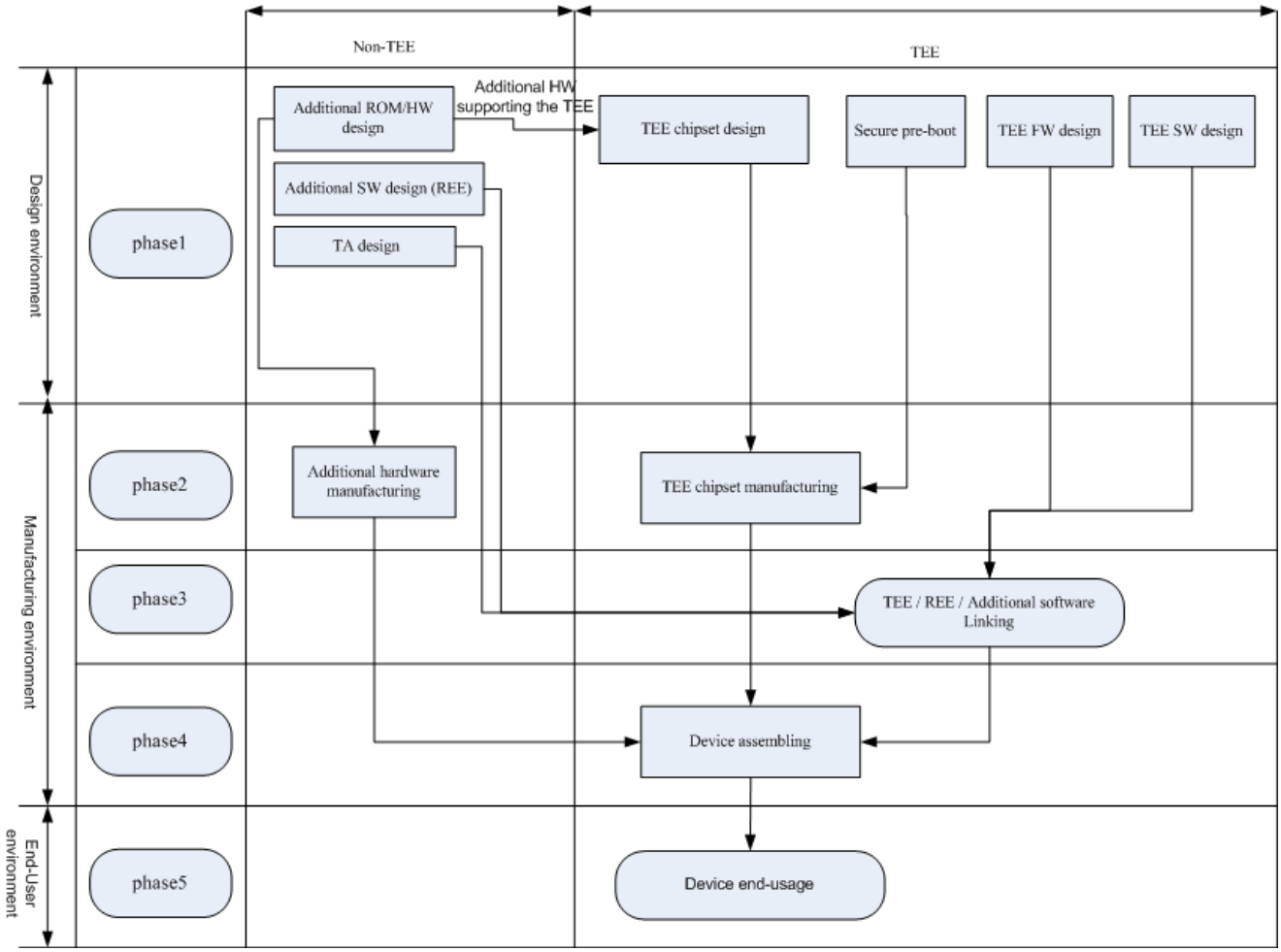

Figure 1-3 TOE's Device Life Cycle

It is split in five phases:

Phase 1: Firmware / Software / Hardware design

The TEE software developer is in charge of the TEE software development and testing compliant with GlobalPlatform specifications. He develops the TEE initialization code that instantiates/initializes the TEE (e.g. part of the secure boot code) and specifies the TEE software linking requirements

The device manufacturer may design additional REE software that will be linked with the TEE in phase 4 to provide REE controlled resources. Both the device manufacturer and TEE software developer may design Trusted Applications that they will integrate in phase 4.

The TEE hardware designer is in charge of designing (part of) the processor(s) where the TEE software runs and designing (part of) the hardware security resources used by the TEE.

The silicon manufacturer designs the ROM code and the secure portion of the TEE chipset.

Phases2: Chipset manufacturing

The silicon manufacturer produces the TEE chipset and enables, sets or seeds the root of trust of the TEE.

Phases3: TEE Software manufacturing

The device manufacturer is responsible for the integration, validation and preparation of the software to load in the product that will include the TEE software, any pre-installed Trusted Application, and additional software required to use the product (e.g. REE, Client Applications).

Phases4: Device manufacturing

The device manufacturer is responsible for the device assembling and initialization and any other operation on the device (including loading or installation of Trusted Applications) before delivery to the end-user.

Phases5: 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.

The TOE operational phase starts in Phase 4.

1.5.1 Involved Actors

Actors	Identification
TEE Software developer	Huawei Central Software Huawei Hisilicon Huawei Mobile Dept
TEE Hardware designer	Huawei Hisilicon
Silicon manufacturer	TSMC Limited ASE Technology Co., Ltd SPIL Co., Ltd
Device manufacturer	Huawei Machine Co., Ltd Shenzhen FuTaiHong Precision Industry Co., Ltd

1.5.2 Involved Sites

Sites	Identification
TEE Software development	Huawei Central Software – Beijing, China Huawei Hisilicon – Shanghai, China

	Huawei Mobile Dept – Shanghai, China
TEE Hardware design	Huawei Hisilicon – Shanghai, China
Silicon manufacturing	TSMC Limited – Taiwan, China ASE Technology Co., Ltd -- Taiwan, China SPIL Co., Ltd -- Taiwan, China
Device manufacturing	Huawei Machine Co., Ltd – Dongguan, China Shenzhen FuTaiHong Precision Industry Co., Ltd – Shenzhen, China

2 Conformance Claims

2.1 CC conformance claim

This Security Target claims to be conformant to CC Part2 extended and CC Part3 extended. The CC Part 2 is extended with the security functional components FCS_RNG.1 Random numbers generation and FPT_INI.1 TSF initialization. The CC Part 3 is extended with the security assurance component AVA_TEE.2 vulnerability analysis. The extended components are defined in chap.5.

Common Criteria [CC] version 3.1 revision 5 is the basis for this conformance claim.

2.2 PP claim

This Security Target claims strict conformance to the GPD_SPE_021 (PP-configuration composed of the base Protection Profile only) of the GlobalPlatform Device Committee TEE Protection Profile Version1.2.1 [TEE PP].

3 Security Problem Definition

3.1 Threats to Security

3.1.1 Assets

The following paragraph presents the assets of the TOE and their properties:

TEE identification

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

Properties: unique and non-modifiable.

Application Note: The TEE identifier is intended to be public and exposed to any software running on the device, not only to Trusted Applications.

RNG

The Random Number Generator.

Properties: unpredictable random numbers, sufficient entropy.

TA code

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

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

TA data and keys

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

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

TA instance time

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

Properties: monotonicity.

TEE runtime data

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

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

TEE persistent data

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

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

TEE firmware

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

Properties: authenticity, integrity.

TEE software initialization code and data

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

Properties: integrity.

TEE storage root of trust

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

Properties: integrity and confidentiality.

3.1.2 Users / Subjects

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

Trusted Application (TA)

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

Rich Execution Environment (REE)

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

3.1.3 Threats

This ST targets threats to the TEE assets that arise during the end-usage phase and can be achieved by software means. Attackers are individuals or organizations with remote or physical (local) access to the device embedding the TEE. The user of the device becomes a potential attacker when the TEE holds assets of third parties. The motivations behind the attacks may be very diverse and in general are linked to the Trusted Application running on the TEE. An attacker may, for instance, try to steal content of the device's owner (such as passwords stored in the device) or content of a service provider, or to unduly benefit from TEE or TA services (such as accessing

corporate network or performing unauthorized use of DRM content either in the same device or in other devices) or to threaten the reputation of the device/TEE manufacturer or service providers. The impact of an attack depends not only on the value of the individual assets attacked, but, in some cases, on the possibility to reproduce the attack rapidly at low cost: single attacks performed on a given TEE-enabled device have lower impact than massive attacks that reach many devices at the same time.

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

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

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

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

Remote attacker: This exploitation profile performs the attack on a remotely-controlled device or alternatively makes a downloadable tool that is very convenient to end-users. The attacker retrieves details of the vulnerability identified in the identification phase and outputs such as attack code/executable provided by the identifier. The attacker then makes a remote tool or malware and uses techniques such as phishing to have it downloaded and executed by a victim, or alternatively makes a friendly tool available on the internet. Note that the design of a new malware, trojan, virus, or rooting tool is often performed from an existing base, available on the internet

Basic device attacker: This exploitation profile has physical access to the target device; it is the end-user or someone on his behalf. The attacker retrieves attack code/application from the identifier, guidelines written on the internet on how to perform the attack, downloads and uses tools to jailbreak/root/reflash the device in order to get privileged access to the REE allowing the execution of the exploit. The attacker may be a layman or have some level of expertise but the attacks do not require any specific equipment.

In all cases, the overall attack potential strongly limits the possibility to face advanced attackers performing the exploits. For large-scale exploitation attacks, we refer to Annex A [TEE PP] for a comprehensive description of the identification and exploitation phases, the applicable attack potential quotation table and a representative set of attacks a TEE may have to face in the field. Due to the somehow more limited interest and possibility of spreading the exploits, attacks against managed devices should only be subset of the attacks in the Annex A.

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

T.ABUSE_FUNCT

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

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

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

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

Application Note:

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

In particular a fake application running in the Rich OS 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 threat is not countered by the TEE alone and must be taken into account in the design of the use case, for instance by using an applicative authenticated communication channel between the client and the TA.

T.CLONE

An attacker manages to copy TEE related data of a first device on a second device and makes this device accept them as genuine data.

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

T.FLASH_DUMP

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

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

Application Note:

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

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

T.IMPERSONATION

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

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

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

T.ROGUE_CODE_EXECUTION

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

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

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

Application Note:

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

T.PERTURBATION

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

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

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

Application Note:

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

T.RAM

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

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

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

Application Note:

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

T.RNG

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

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

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

T.SPY

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

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

Application Note:

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

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

T.TEE_FIRMWARE_DOWNGRADE

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

Assets threatened directly (integrity): TEE firmware.

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

T.STORAGE_CORRUPTION

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

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

Application Note:

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

3.2 OSPs

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

OSP.INTEGRATION_CONFIGURATION

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

OSP.SECRETS

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

3.3 Assumptions

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

A.PROTECTION_AFTER_DELIVERY

It is assumed that the TOE is protected by the environment after delivery and before entering the final usage phase. It is assumed that the persons manipulating the TOE in the operational environment apply the TEE guidelines (e.g. user and administrator guidance, installation documentation, personalization guide). It is also assumed that the

persons responsible for the application of the procedures contained in the guides, and the persons involved in delivery and protection of the product have the required skills and are aware of the security issues.

A.ROLLBACK

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

A.TA_DEVELOPMENT

TA developers are assumed to be competent and trustworthy individuals who will comply with the TA development guidelines set by the TEE provider. In particular, TA developers are assumed to consider the following principles during the development of the Trusted Applications:

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

4 Security Objectives

4.1 Security Objectives for the Operational Environment

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

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

OE.INTEGRATION_CONFIGURATION

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

The TOE shall be installed and configured correct to ensure the TOE running in a secure state. The process of TEE identifier generating shall ensure statistical uniqueness of the TEE identifier.

OE.PROTECTION_AFTER_DELIVERY

The TOE shall be protected by the environment after delivery and before entering the final usage phase. The persons manipulating the TOE in the operational environment shall apply the TEE guidance (e.g. user and administrator guidance, installation documentation, personalization guide).

The persons responsible for the application of the procedures contained in the guides, and the persons involved in delivery and protection of the product have the required skills and are aware of the security issues.

Application Note:

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

OE.ROLLBACK

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

OE.SECRETS

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

OE.TA_DEVELOPMENT

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

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

4.2 Security Objectives for the TOE

The following objectives must be met by the TOE:

O.CA_TA_IDENTIFICATION

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

Application Note:

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

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

O.KEYS_USAGE

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

O.TEE_ID

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

Application Note:

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

O.INITIALIZATION

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

- the integrity of the TEE initialization code and data used to load the TEE firmware
- the authenticity of the TEE firmware
- and that the TEE is bound to the SoC of the device. In particular, the TEE shall protect the TEE firmware against downgrade attacks.

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 (cf. [SA]).

O.INSTANCE_TIME

The TEE 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.

O.OPERATION

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

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

Application Note:

- Programmer errors or violation of good practices (e.g. that exploit multi-threading or context/session management) might become attack-enablers. The REE may be harmful but < the implementation (TEE) still guarantees the stability and security of TEE > (cf. [CAPI]). In any case, a Trusted Application MUST NOT be able to use a programmer error on purpose to circumvent the security boundaries enforced by an implementation (cf. [IAPI] and [SA])
- Entry points (cf. [SA]): Software in the REE must not be able to call directly to TEE Functions or Trusted Core Framework. The REE software must go through protocols such that the Trusted OS or Trusted Application performs the verification of the acceptability of the operation that the REE software has requested.

O.RNG

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

Application Note:

Random number generation may combine hardware and/or software mechanisms. The RNG functionality is also used for generation of the TEE identifier if this identifier is not generated outside the TEE.

O.RUNTIME_CONFIDENTIALITY

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

- The TEE shall not export any sensitive data, random numbers or secret keys to the REE
- The TEE shall grant access to sensitive data, random numbers or secret keys only to authorized TAs

- The TEE shall clean up sensitive resources as soon as it can determine that their values are no longer needed.

O.RUNTIME_INTEGRITY

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

O.TA_AUTHENTICITY

The TEE shall verify code authenticity of Trusted Applications.

Application Note:

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

O.TA_ISOLATION

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

Application Note:

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

O.TEE_DATA_PROTECTION

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

O.TEE_ISOLATION

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

Application Note:

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

O.TRUSTED_STORAGE

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

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

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

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

Table 4-1 Assets store location

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

4.3 Security Objectives Rationale

4.3.1 Tracing between security objectives and SPD

Table 4-2 Tracing of SPD to Security Objectives

Threats	Security Objectives
T.ABUSE_FUNCT	O.INITIALIZATION, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TEE_ISOLATION, OE.TA_DEVELOPMENT, O.KEYS_USAGE, O.TA_AUTHENTICITY
T.CLONE	O.TEE_ID, O.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, O.RNG
T.FLASH_DUMP	O.TRUSTED_STORAGE
T.IMPERSONATION	O.CA_TA_IDENTIFICATION, O.OPERATION, O.RUNTIME_INTEGRITY,
T.ROGUE_CODE_EXECUTION	O.OPERATION,

Threats	Security Objectives
	O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY, O.TA_AUTHENTICITY, O.INITIALIZATION
T.PERTURBATION	O.INITIALIZATION, O.INSTANCE_TIME, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TA_PERSISTENT_TIME, O.TEE_DATA_PROTECTION, O.TEE_ISOLATION, O.TA_AUTHENTICITY
T.RAM	O.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION
T.RNG	O.INITIALIZATION,O.RNG O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY
T.SPY	O.RUNTIME_CONFIDENTIALITY, O.TA_ISOLATION, O.TEE_ISOLATION, O.TRUSTED_STORAGE
T.TEE_FIRMWARE_DOWNGRADE	O.INITIALIZATION, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY
T.STORAGE_CORRUPTION	O.OPERATION, O.ROLLBACK_PROTECTION, OE.ROLLBACK, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, O.TA_AUTHENTICITY, O.INITIALIZATION

Threats	Security Objectives
OSP.INTEGRATION_CONFIGURATION	OE.INTEGRATION_CONFIGURATION
OSP.SECRETS	OE.SECRETS
A.PROTECTION_AFTER_DELIVERY	OE.PROTECTION_AFTER_DELIVERY
A.ROLLBACK	OE.ROLLBACK
A.TA_DEVELOPMENT	OE.TA_DEVELOPMENT

Table 4-3 Tracing of Security Objectives to SPD

Security Objectives	SPD
O.CA_TA_IDENTIFICATION	T.IMPERSONATION
O.KEYS_USAGE	T.ABUSE_FUNCT
O.TEE_ID	T.CLONE
O.INITIALIZATION	T.ABUSE_FUNCT, T.CLONE, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.RAM, T.RNG, T.TEE_FIRMWARE_DOWNGRADE, T.STORAGE_CORRUPTION
O.INSTANCE_TIME	T.PERTURBATION
O.OPERATION	T.ABUSE_FUNCT, T.IMPERSONATION, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.STORAGE_CORRUPTION
O.RNG	T.CLONE, T.RNG
O.RUNTIME_CONFIDENTIALITY	T.ABUSE_FUNCT, T.CLONE, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.RAM, T.RNG, T.SPY
O.RUNTIME_INTEGRITY	T.ABUSE_FUNCT, T.CLONE, T.IMPERSONATION, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.RAM, T.RNG
O.TA_AUTHENTICITY	T.ABUSE_FUNCT, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.STORAGE_CORRUPTION

Security Objectives	SPD
O.TA_ISOLATION	T.PERTURBATION, T.RAM, T.SPY
O.TEE_DATA_PROTECTION	T.ABUSE_FUNCT, T.CLONE, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.STORAGE_CORRUPTION
O.TEE_ISOLATION	T.ABUSE_FUNCT, T.PERTURBATION, T.RAM, T.SPY
O.TRUSTED_STORAGE	T.CLONE, T.FLASH_DUMP, T.ROGUE_CODE_EXECUTION, T.SPY, T.STORAGE_CORRUPTION
OE.INTEGRATION_CONFIGURATION	T.ROGUE_CODE_EXECUTION, T.TEE_FIRMWARE_DOWNGRADE OSP.INTEGRATION_CONFIGURATION
OE.PROTECTION_AFTER_DELIVERY	T.ROGUE_CODE_EXECUTION, T.TEE_FIRMWARE_DOWNGRADE A.PROTECTION_AFTER_DELILVERY
OE.ROLLBACK	T.STORAGE_CORRUPTION A.ROLLBACK
OE.SECRETS	OSP.SECRETS
OE.TA_DEVELOPMENT	T.ABUSE_FUNCT A.TA_DEVELOPMENT

4.3.2 Justification of tracing

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.

Table 4-4 Threat Rationale

Threat	Security Objectives
T.ABUSE_FUNCT	The combination of the following objectives ensures protection against abuse of functionality: O.INITIALIZATION ensures that the TEE security functionality is correctly initialized O.OPERATION ensures correct operation of the security functionality and a proper management of failures O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data O.RUNTIME_INTEGRITY ensures protection against unauthorized

Threat	Security Objectives
	modification of security functionality at runtime O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified O.TEE_DATA_PROTECTION ensures that the data used by the TEE are authentic and consistent O.TEE_ISOLATION enforces the separation between the TEE and the outside (REE and TAs) O.KEYS_USAGE controls the usage of cryptographic keys OE.TA_DEVELOPMENT enforces TA development principles, which are meant in particular to prevent disclosing information or performing modifications upon request of unauthorized entities.
T.CLONE	The combination of the following objectives ensures protection against cloning: O.TEE_ID provides the unique TEE identification means O.INITIALIZATION ensures that the TEE is bound to the SoC of the device O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data, in particular TSF data used to bind the TEE to the device O.RUNTIME_INTEGRITY prevents against unauthorized modification at runtime of security functionalities or data used to detect or prevent cloning O.TEE_DATA_PROTECTION prevents the TEE from using TEE data that is inconsistent or not authentic O.TRUSTED_STORAGE ensures that the trusted storage is bound to the device and prevents the TEE from using data that is inconsistent or not authentic O.RNG ensures that the TEE identifier is in fact unique when generated inside the TOE
T.FLASH_DUMP	O.TRUSTED_STORAGE ensures the confidentiality of the data stored in external memory.
T.IMPERSONATION	The combination of the following objectives ensures protection against application impersonation attacks: O.CA_TA_IDENTIFICATION ensures the protection of Client identities and the possibility to distinguish Client Applications and Trusted Applications O.OPERATION ensures the verification of Client identities before any operation on their behalf O.RUNTIME_INTEGRITY prevents against unauthorized modification of security functionality at runtime.
T.ROGUE_CODE_EXECUTION	The combination of the following objectives ensures protection against import of malicious code: O.INITIALIZATION ensures that the TEE security functionality is correctly initialized and the integrity of TEE firmware O.OPERATION ensures correct operation of the security functionality

Threat	Security Objectives
	O.RUNTIME_CONFIDENTIALITY covers runtime TEE data which might influence the behavior of the TEE O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime O.TEE_DATA_PROTECTION covers persistent TEE data which might influence the behavior of the TEE O.TRUSTED_STORAGE ensures protection of the storage from which code might be imported OE.INTEGRATION_CONFIGURATION covers the import of foreign code in a phase other than the end-user phase OE.PROTECTION_AFTER_DELIVERY covers the import of foreign code in a phase other than the end-user phase.
T.PERTURBATION	The combination of the following objectives ensures protection against perturbation attacks: O.INITIALIZATION ensures that the TEE security functionality is correctly initialized O.INSTANCE_TIME ensures the reliability of instance time stamps O.OPERATION ensures correct operation of the security functionality and a proper management of failures O.RUNTIME_CONFIDENTIALITY covers runtime TEE data which might influence the behavior of the TEE O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime O.TA_ISOLATION ensures the separation of the TA O.TEE_DATA_PROTECTION covers persistent TEE data which might influence the behavior of the TEE O.TEE_ISOLATION enforces the separation between the TEE and the outside (REE and TAs).
T.RAM	The combination of the following objectives ensures protection against RAM attacks: O.INITIALIZATION ensures that the TEE security functionality is correctly initialized and that the initialization process is protected from the REE O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data at runtime O.RUNTIME_INTEGRITY protects against unauthorized modification of code and data at runtime O.TA_ISOLATION provides a memory barrier between TAs O.TEE_ISOLATION provides a memory barrier between the TEE and the REE.

Threat	Security Objectives
T.RNG	The combination of the following objectives ensures protection of the random number generation: O.INITIALIZATION ensures the correct initialization of the TEE security functions, in particular the RNG O.RNG ensures that random numbers are unpredictable, have sufficient entropy and are not disclosed O.RUNTIME_CONFIDENTIALITY ensures that confidential data is not disclosed O.RUNTIME_INTEGRITY protects against unauthorized modification, for instance to force the output of the RNG.
T.SPY	The combination of the following objectives ensures protection against disclosure: O.RUNTIME_CONFIDENTIALITY ensures protection of confidential data at runtime O.TA_ISOLATION ensures the separation between TAs O.TEE_ISOLATION ensures that neither REE nor TAs can access TEE data O.TRUSTED_STORAGE ensures that data stored in the trusted storage locations is accessible by the TA owner only.
T.TEE_FIRMWARE_DOWNGRADE	The combination of the following objectives ensures protection against TEE firmware downgrade: O.INITIALIZATION ensures that the firmware that is executed is the version that was intended OE.INTEGRATION_CONFIGURATION ensures that the firmware installed in the device is the version that was intended OE.PROTECTION_AFTER_DELIVERY ensures that the firmware has not been modified after delivery.
T.STORAGE_CORRUPTION	The combination of the following objectives ensures protection against corruption of non-volatile storage: O.OPERATION ensures the correct operation of the TEE security functionality, including storage O.TEE_DATA_PROTECTION ensures that stored TEE data are genuine and consistent O.TRUSTED_STORAGE enforces detection of corruption of the TA's storage O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified O.INITIALIZATION 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.

Table 4-5 Assumption Rationale

Assumption	Security Objectives
A.PROTECTION_AFTER_DELIVERY	OE.PROTECTION_AFTER_DELIVERY directly covers this assumption.
A.ROLLBACK	OE.ROLLBACK directly covers this assumption.
A.TA_DEVELOPMENT	OE.TA_DEVELOPMENT directly covers this assumption.

Table 4-6 OSP Rationale

OSP	Security Objectives
OSP.INTEGRATION_CONFIGURATION	OE.INTEGRATION_CONFIGURATION directly covers this OSP.
OSP.SECRETS	OE.SECRETS directly covers this OSP.

5 Extended Components Definition

5.1 Extended Family FCS_RNG - Generation of random numbers

5.1.1 Description

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

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

5.1.2 Family behavior

Description

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

Hierarchical to: No other components.

Management: No management activities are foreseen.

Audit

No actions are defined to be auditable.

Definition

FCS_RNG.1 Random numbers generation

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

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

Dependencies: No dependencies.

5.2 Extended Family FPT_INI - TSF initialisation

5.2.1 Description

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

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

5.2.2 Family behavior

Description

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

Hierarchical to: No other components.

Management: No management activities are foreseen.

Audit: No actions are defined to be auditable.

Definition

FPT_INI.1 TSF initialisation

FPT_INI.1.1 The TOE initialization function shall verify

- the integrity of TEE initialization code and data
- the authenticity and integrity of TEE firmware
- the integrity of the storage root of trust
- the integrity of the TEE identification data
- the version of the firmware to prevent downgrade to previous versions
- [assignment: list of implementation-dependent verifications]

prior to establishing the TSF in a secure initial state.

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

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

Dependencies: No dependencies.

5.3 Extended Class AVA_TEE - Vulnerability analysis of TEE

5.3.1 Description

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

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

The family 'Vulnerability analysis of TEE (AVA_TEE)' defines requirements for evaluator independent vulnerability search and penetration testing of TEE. The main characteristic of the new family, which is almost identical to AVA_VAN, is to introduce the TEE-specific attack potential scale defined in Annex A.1. This annex also provides the TEE attack potential calculation table and a catalogue of TEE-specific attack methods. In this current version of the Protection Profile, only one level of the TEE-specific attack potential scale, namely TEE-Low, is used, in the component AVA_TEE.2.

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

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

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

5.3.2 Class behavior

Description

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

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

Definition

AVA_TEE.2 TEE vulnerability analysis

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

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

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

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

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

AVA_TEE.2.4E The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing TEE-Low attack potential.

Dependencies: ADV_ARC.1 Security architecture description

ADV_FSP.2 Security-enforcing functional specification

ADV_TDS.1 Basic design

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

6 Security Requirements

6.1 Conventions

The conventions used in descriptions of the SFRs are as follows:

(1) Assignment: Indicated with *italicized* text;

(2) Refinement: Indicated with **bold text** and strikethroughs, if necessary;

(3) Selection: Indicated with underlined text;

(4) Assignment within a Selection: Indicated with *italicized and underlined text*;

(5) Iteration: Indicated by appending the iteration number in parenthesis, e.g. (1), (2), (3) and /or by adding a string starting with "/";

(6) Application Notes added by the ST author are called 'Additional Application Note' which are enumerated as 'a', 'b', ... and are formatted with underline such as "Additional Application Note a: Audit shall be enabled by default";

(7) References: Indicated with [square brackets].

6.2 Definition of security policies

The statement of the security functional requirements rely on the following characterization of the TEE in terms of users, subjects, objects, information, user data, TSF data, operations and their security attributes (cf. CC Part 1 [CC1] for the definition of these notions).

Users stand for entities outside the TOE:

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

Subjects stand for active entities inside the TOE:

- S.TA_INSTANCE: any TA instance with security attribute "TA_identity" (TA identifier)
- S.TA_INSTANCE_SESSION: any session within a given TA instance, with security attribute "client_identity" (CA identifier)

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

Objects stand for passive entities inside the TOE:

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

Cryptographic objects are a special kind of TEE objects:

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

Information stands for data exchanged between subjects:

- I.RUNTIME_DATA (user data or TSF Data depending on the owner): data belonging to the TA or to the TEE itself. Stands for parameter values, return values, content of memory regions in clear text. Note: Data that is encrypted and authenticated is not considered I.RUNTIME_DATA.

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

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

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

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

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

Other operations:

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

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

Runtime Data Information Flow Control SFP:

- Purpose: To control the flow of runtime data from and to executable entities and memory. This policy contributes to ensure the integrity and confidentiality of runtime data
- Subjects: S.TA_INSTANCE, S.TA_INSTANCE_SESSION, S.API, S.COMM_AGENT, S.RESOURCE, S.RAM_UNIT
- Information: I.RUNTIME_DATA
- Security attributes: S.RESOURCE.state, S.RAM_UNIT.rights and S.API.caller

- SFR instances: FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_ITT.1/Runtime.

TA Keys Access Control SFP:

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

Trusted Storage Access Control SFP:

- Purpose: To control the access to TA storage where persistent TA data and keys are stored, which is granted on behalf of the owner TA only. This policy also enforces the binding of TA trusted storage to the TEE storage root of trust OB.SRT
- Subjects: S.API
- Objects: OB.TA_STORAGE, OB.SRT
- Security attributes: S.API.caller, OB.TA_STORAGE.owner, OB.TA_STORAGE.inExtMem, OB.TA_STORAGE.TEE_identity and OB.SRT.TEE_identity
- Operations: OP.LOAD, OP.STORE
- SFR intances: FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FMT_SMF.1.

6.3 Security Functional Requirements

6.3.1 Identification

FIA_ATD.1 User attribute definition

FIA_ATD.1.1 The TSF shall maintain the following list of security attributes belonging to individual users: **CA_identity, TA_identity, TA_properties.**

Application Note:

The lifespan of the attributes in such a list is the following:

CA_identity: The lifetime of this attribute is that of the lifetime of the client session to the TA

TA_identity: The availability of this attribute is that of the availability of the TA to clients, limited further by the TAs presence in the system

TA_properties: The lifetime of this attribute is that of the availability of the TA to clients, limited further by the TAs presence in the system.

FIA_UID.2 User identification before any action

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

Application Note:

User stands for Client Application or Trusted Application.

FIA_USB.1 User-subject binding

FIA_USB.1.1 The TSF shall associate the following user security attributes with subjects acting on the behalf of that user:

- **Client (CA or TA) identity is codified into the client_identity of the requested TA session**

FIA_USB.1.2 The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users:

- **If the client is a TA, then the client_identity must be equal to the TA_identity of the TA subject that is the client**
- **If the client is a CA，then the client identity must indicate the CA client.**

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

- **No modification of client_identity is allowed after initialization**

Application Note:

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

FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles

- **TSF**
- **TA_User**

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

Application Note:

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

6.3.2 Confidentiality, Integrity and Isolation

FDP_IFC.2/Runtime Complete information flow control

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

- **Subjects: S.TA_INSTANCE, S.TA_INSTANCE_SESSION, S.API, S.COMM_AGENT, S.RESOURCE, S.RAM_UNIT**
- **Information: I.RUNTIME_DATA**

and all operations that cause that information to flow to and from subjects covered by the SFP.

FDP_IFC.2.2/Runtime The TSF shall ensure that all operations that cause any information in the TOE to flow to and from any subject in the TOE are covered by an information flow control SFP.

Application Note:

The flow control policy specifies the conditions to communicate runtime data from one subject to another. It applies to operations that are standard interfaces of these subjects.

FDP_IFF.1/Runtime Simple security attributes

FDP_IFF.1.1/Runtime The TSF shall enforce the Runtime Data Information Flow Control SFP based on the following types of subject and information security attributes: S.RESOURCE.state, S.RAM_UNIT.rights and S.API.caller.

FDP_IFF.1.2/Runtime The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

- **Rules for information flow between S.TA_INSTANCE and S.RAM_UNIT:**
  - Flow of I.RUNTIME_DATA from S.TA_INSTANCE to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(S.TA_INSTANCE) is Write or ReadWrite
  - Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.TA_INSTANCE is allowed only if S.RAM_UNIT.rights(S.TA_INSTANCE) is Read or ReadWrite
- **Rules for information flow from and to S.COMM_AGENT:**
  - Flow of I.RUNTIME_DATA from S.COMM_AGENT to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(REE) is Write or ReadWrite
  - Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.COMM_AGENT is allowed only if S.RAM_UNIT.rights(REE) is Read or ReadWrite
- **Rules for information flow from and to S.API:**
  - Flow of I.RUNTIME_DATA from S.API to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(S.API.caller) is Write or ReadWrite
  - Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.API is allowed only if S.RAM_UNIT.rights(S.API.caller) is Read or ReadWrite
- **Rules for information flow from and to S.RESOURCE:**
  - Flow of I.RUNTIME_DATA between S.API and S.RESOURCE is allowed only if the resource is under TEE control (S.RESOURCE.state = TEE).

FDP_IFF.1.3/Runtime The TSF shall enforce **none**

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

- **Rules for information flow from and to S.TA_INSTANCE_SESSION:**
  - Flow of I.RUNTIME_DATA that are parameter or return values is allowed between S.TA_INSTANCE_SESSION and S.COMM_AGENT
  - Flow of I.RUNTIME_DATA that are parameter or return values is allowed between S.TA_INSTANCE_SESSION and S.API.

FDP_IFF.1.5/Runtime The TSF shall explicitly deny an information flow based on the following rules: Any information flow involving a TEE subject unless one of the conditions stated in FDP_IFF.1.1/1.2/1.3/1.4 holds.

Application Note:

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

FDP_ITT.1/Runtime Basic internal transfer protection

FDP_ITT.1.1/Runtime The TSF shall enforce the Runtime Data Information Flow Control SFP to prevent the disclosure and modification of user data when it is transmitted between physically separated parts of the TOE.

Application Note:

The resources used by the TEE may reside in "physically separated parts". This requirement addresses data transmission through communication buses (recall that the definition of S.RESOURCES does not include the buses).

FDP_RIP.1/Runtime Subset residual information protection

FDP_RIP.1.1/Runtime The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **TEE and TA runtime objects.**

Application Note:

This operation applies in particular upon:

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

FPT_ITT.1/Runtime Basic internal TSF data transfer protection

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

Application Note:

The resources used by the TEE may reside in "physically separated parts".

6.3.3 Cryptography

FCS_COP.1 Cryptographic operation - *Encryption/Decryption*

FCS_COP.1/ Encryption/Decryption The TSF shall perform - Symmetric *Encryption/Decryption* in accordance with a specified cryptographic algorithm [AES] with supported modes AES-XTS that meet the following standards FIPS 197, IEEE Std 1619-2007 with cryptographic key sizes ***256***

FCS_COP.1 Cryptographic operation - Sign/Verify

FCS_COP.1/ Sign/Verify The TSF shall perform **Sign/Verify** in accordance with a specified cryptographic algorithm **RSA** that meet the following standards **PKCS #1** with cryptographic key sizes ***2048***

FCS_COP.1 Cryptographic operation - Hashing

FCS_COP.1/ Hashing The TSF shall perform **Hashing** in accordance with a specified cryptographic algorithm **Hash** that meet the following standards **FIPS 180-4** with supported mode **SHA256**.

FCS_COP.1 Cryptographic operation - HMAC

FCS_COP.1/ HMAC The TSF shall perform **HMAC** in accordance with a specified cryptographic algorithm **HMAC** that meet the following standards **RFC 4231** with supported mode **HMAC-SHA-256** with cryptographic key size ***256***

FCS_COP.1 Cryptographic operation - CMAC

FCS_COP.1/ CMAC The TSF shall perform **CMAC** in accordance with a specified cryptographic algorithm **CMAC** that meet the following standards **NIST SP800-38B** with cryptographic key size ***256***

Application Note:

This SFR cryptographic operations used within the TOE for:

- verifying the authenticity of firmware and software of the TOE using RSA2048 (**FCS_COP.1/ Sign/Verify**) and SHA256 (**FCS_COP.1/ Hashing)**
- verifying the authenticity of TA Code, using RSA2048 (**FCS_COP.1/ Sign/Verify**) and SHA256 (**FCS_COP.1/ Hashing)**
- protecting the consistency and confidentiality of Trusted Storage data. These operations are based on the TEE storage root of trust key. Using HMAC (**FCS_COP.1/ HMAC**) , AES_XTS_256 (**FCS_COP.1/ Encryption/Decryption**) and CMAC (**FCS_COP.1/ KDF) .**

FDP_ACC.1/TA_keys Subset access control

FDP_ACC.1.1/TA_keys The TSF shall enforce the TA Keys Access Control SFP on

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

FDP_ACF.1/TA_keys Security attribute based access control

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

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

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

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

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

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

Application Note:

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

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

FMT_MSA.1/TA_keys Management of security attributes

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

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

FMT_MSA.3/TA_keys Static attribute initialization

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

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

6.3.4 Initialization, Operation and Firmware Integrity

FAU_ARP.1 Security alarms

FAU_ARP.1.1 The TSF shall ***enter secure state*** upon detection of a potential security violation.

Refinement:

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

- detection of consistency violation of TA data, TA code or TEE data: ***reject the error or malicious data*****.**
- detection of TEE firmware integrity violation**: *iTrustee will stop booting thus boot failed***.

Application Note:

The TOE implement several security enhancement such as Code segment Read Only, Data segment Never eXecute, Stack Canary, CFI, ASLR etc. to detect the consistency violation of TA data, TA code or TEE data. When the TOE detect potential security violation, TSF will enter secure state, stop the module running where failure occurred.

FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for **integrity errors** on all objects, based on the following attributes: **user data attributes.**

Refinement:

The TSF shall monitor TEE runtime data, TEE persistent data, TA data and keys and TA code stored in containers controlled by the TSF for authenticity and consistency errors on all objects, based on the following attributes: 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 ***take secure action that does not depend on the compromised data***.

Refinement:

- Upon detection of authenticity or consistency errors in TEE runtime data or TEE persistent data, the TSF shall ***take secure action that does not depend on the compromised data***
- Upon detection of TA code authenticity or consistency errors, the TSF shall **abort the execution of the TA instance**

- Upon detection of TA data or TA keys authenticity or consistency errors, the TSF shall
  - Not give back any compromised data
  - take secure action that does not depend on the compromised data.

Application Note:

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

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

FPT_FLS.1 Failure with preservation of secure state

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

- **Device binding failure**
- **Cryptographic operation failure**
- **Invalid CA requests, in particular bad-formed requests**
- **Panic states**
- **TA code, TA data or TA keys authenticity or consistency failure**
- **TEE data (in particular TA properties, TEE keys and all security attributes) authenticity or consistency failure**
- **TEE initialization failure**
- **Unexpected commands in the current TEE state**

Application Note:

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

If everything goes well since TOE start, the TOE works in normal state. When failure occurred, the TOE stop the failure module running so that the TOE will not leak any sensitive information like encryption key, key structure data to REE world.

FPT_INI.1 TSF initialization

FPT_INI.1.1 The TOE initialization function shall verify

- **the integrity of TEE initialization code and data**
- **the authenticity and integrity of TEE firmware**
- **the integrity of the storage root of trust**
- **the integrity of the TEE identification data**
- **the version of the firmware to prevent downgrade to previous versions**

prior to establishing the TSF in a secure initial state.

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

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

FMT_SMF.1 Specification of Management Functions

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

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

FPT_TEE.1 Testing of external entities

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

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

6.3.5 TEE Identification

FAU_SAR.1 Audit review

FAU_SAR.1.1 The TSF shall provide **all users** with the capability to read **TEE identifier** from the audit records.

FAU_SAR.1.2 The TSF shall provide the audit records in a manner suitable for the user to interpret the information.

FAU_STG.1 Protected audit trail storage

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

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

Application Note:

The TEE identifier is generated on-TEE, and the generating algorithm is defined by the manufacturer who will guarantee the TEE identifier to be unique.

When the TOE startup, TOE will read DIEID and HUK from Efuse, and generate the TEE identifier based on these two parameters. The TEE identifier is stored in TEE's volatile memory, and will be lost when TOE shutdown.

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

6.3.6 Instance Time

FPT_STM.1/Instance time Reliable time stamps

FPT_STM.1.1/Instance time The TSF shall be able to provide reliable time stamps.

Refinement:

The TSF shall be able to provide time stamps to TA instances such that time stamps are monotonic during the TA instance lifetime.

Application Note:

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

6.3.7 Random Number Generator

FCS_RNG.1 Random numbers generation

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

- *CPG.1: Total failure test of the entropy source.*
- *CPG.2: Online test of the raw random number sequence.*
- *CPG.3: Online test of the internal random numbers.*

FCS_RNG.1.2 The TSF shall provide random numbers that meet **Test T0 to Test T8** defined in **AIS31**.

6.3.8 Trusted Storage

FDP_ACC.1/Trusted Storage Subset access control

FDP_ACC.1.1/Trusted Storage The TSF shall enforce the Trusted Storage Access Control SFP on

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

FDP_ACF.1/Trusted Storage Security attribute based access control

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

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

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

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

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

- **Any access to a trusted storage attempted from S.API without valid caller (S.API.caller = undefined)**

- **Any access to a trusted storage that was bound to a different TEE (OB.TA_STORAGE.TEE_identity different from OB.SRT.TEE_identity)**
- **Any access to a trusted storage from a subject different from S.API**

FDP_ROL.1/Trusted Storage Basic rollback

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

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

Application Note:

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

FMT_MSA.1/Trusted Storage Management of security attributes

FMT_MSA.1.1/Trusted Storage The TSF shall enforce the Trusted Storage Access Control SFP to restrict the ability to query the security attributes OB.TA_STORAGE.owner, OB.TA_STORAGE.inExtMem, OB.TA_STORAGE.TEE_identity and OB.SRT.TEE_identity to TA_User role.

FMT_MSA.3/Trusted Storage Static attribute initialization

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

FMT_MSA.3.2/Trusted Storage The TSF shall allow the **TA_User** to specify alternative initial values to override the default values when an object or information is created.

FDP_ITT.1/Trusted Storage Basic internal transfer protection

FDP_ITT.1.1/Trusted Storage The TSF shall enforce the **Trusted Storage Access Control SFP** to prevent the **disclosure and modification** of user data when it is transmitted between physically separated parts of the TOE.

6.4 Security Assurance Requirements

This ST follows a set of Security Assurance Requirements (SARs) which provided by [TEE PP] for the TOE, it consists of the EAL 2 package augmented with the extended AVA_TEE.2 SAR. This SAR raises the AVA_VAN.2 Basic attack potential to a TEE-Low attack potential.

6.5 Security Requirements Rationale

6.5.1 SFR Necessity and Sufficiency Analysis

Table 6-1 Security Objectives and SFRs - Coverage

Security Objectives	Security Functional Requirements
O.CA_TA_IDENTIFICATION	FIA_ATD.1, FIA_UID.2, FIA_USB.1
O.KEYS_USAGE	FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys, FMT_SMF.1, FCS_COP.1, FMT_SMR.1
OE.TEE_ID	FAU_SAR.1, FCS_RNG.1, FPT_INI.1, FAU_STG.1
O.INITIALIZATION	FPT_FLS.1, FPT_INI.1, FCS_COP.1
O.INSTANCE_TIME	FPT_STM.1/Instance time
O.OPERATION	FAU_ARP.1, FDP_SDI.2, FIA_ATD.1, FIA_UID.2, FIA_USB.1, FMT_SMR.1, FPT_FLS.1, FDP_SDI.2/Rollback, FPT_FLS.1/Rollback, FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FMT_SMF.1, FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys
O.RNG	FCS_RNG.1
O.RUNTIME_CONFIDENTIALITY	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_ITT.1/Runtime, FDP_RIP.1/Runtime,

Security Objectives	Security Functional Requirements
	FPT_ITT.1/Runtime
O.RUNTIME_INTEGRITY	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_ITT.1/Runtime, FPT_ITT.1/Runtime, FDP_SDI.2
O.TA_AUTHENTICITY	FDP_SDI.2, FCS_COP.1, FPT_TEE.1
O.TA_ISOLATION	FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FMT_SMF.1, FCS_COP.1, FPT_FLS.1
O.TEE_DATA_PROTECTION	FDP_SDI.2, FCS_COP.1, FPT_ITT.1/Runtime
O.TEE_ISOLATION	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime
O.TRUSTED_STORAGE	FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FDP_SDI.2, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FCS_COP.1, FPT_INI.1, FMT_SMF.1, FPT_FLS.1, FDP_ITT.1/Trusted Storage

Table 6-2 SFRs and Security Objectives

Security Functional Requirements	Objectives
FIA_ATD.1	O.CA_TA_IDENTIFICATION, O.OPERATION
FIA_UID.2	O.CA_TA_IDENTIFICATION, O.OPERATION
FIA_USB.1	O.CA_TA_IDENTIFICATION, O.OPERATION
FMT_SMR.1	O.KEYS_USAGE, O.OPERATION
FDP_IFC.2/Runtime	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION

Security Functional Requirements	Objectives
FDP_IFF.1/Runtime	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION
FDP_ITT.1/Runtime	O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY
FDP_RIP.1/Runtime	O.RUNTIME_CONFIDENTIALITY
FPT_ITT.1/Runtime	O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION
FCS_COP.1	O.KEYS_USAGE, O.INITIALIZATION, O.TA_AUTHENTICITY, O.TA_ISOLATION, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE
FDP_ACC.1/TA_keys	O.KEYS_USAGE, O.OPERATION
FDP_ACF.1/TA_keys	O.KEYS_USAGE, O.OPERATION
FMT_MSA.1/TA_keys	O.KEYS_USAGE, O.OPERATION
FMT_MSA.3/TA_keys	O.KEYS_USAGE, O.OPERATION
FAU_ARP.1	O.OPERATION
FDP_SDI.2	O.OPERATION, O.RUNTIME_INTEGRITY, O.TA_AUTHENTICITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE
FPT_FLS.1	O.INITIALIZATION, O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FPT_INI.1	O.TEE_ID, O.INITIALIZATION, O.TRUSTED_STORAGE
FMT_SMF.1	O.KEYS_USAGE, O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FPT_TEE.1	O.TA_AUTHENTICITY
FAU_SAR.1	OE.TEE_ID
FAU_STG.1	OE.TEE_ID

Security Functional Requirements	Objectives
FPT_STM.1/Instance time	O.INSTANCE_TIME
FCS_RNG.1	O.TEE_ID, O.RNG
FDP_ACC.1/Trusted Storage	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FDP_ACF.1/Trusted Storage	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FDP_ROL.1/Trusted Storage	O.TRUSTED_STORAGE
FMT_MSA.1/Trusted Storage	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
sted Storage	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FDP_ITT.1/Trusted Storage	O.TRUSTED_STORAGE

Table 6-3 Justification of tracing

Security Objective	Rationale
O.CA_TA_IDENTIFICATION	The following requirements contribute to fulfill the objective: **FIA_ATD.1** enforces the management of the Client and TA identity and properties as security attributes, which then become TSF data, protected in integrity and confidentiality **FIA_UID.2** requires the identification of Client application or TA before any action, thus allowing the access to services and data to authorized users only **FIA_USB.1** enforces the association of the user identity to the active entity that acts on behalf of the user and to check that this is a valid identity.
O.KEYS_USAGE	The following requirements contribute to fulfill the objective: **FCS_COP.1** allows to specify the cryptographic operations in the scope of the evaluation if any **FDP_ACC.1/TA_keys**, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys, FMT_SMR.1 and FMT_SMF.1 state the key access policy, which grants access to the owner of the key only.
O.TEE_ID	The following requirements contribute to fulfill the objective: **FAU_SAR.1** enforces TEE identifier access capabilities **FAU_STG.1** enforces TEE identifier storage capabilities **FPT_INI.1** enforces the integrity of TEE identification, and it states the behavior in case of failure **FCS_RNG.1** enforces statistical uniqueness of the TEE identification data if it is generated on the TOE.

Security Objective	Rationale
O.INITIALIZATION	The following requirements contribute to fulfill the objective: **FPT_FLS.1** states that the TEE has to reach a secure state upon initialization or device binding failure **FCS_COP.1** states the cryptography used to verify the authenticity of TEE firmware **FPT_INI.1** enforces the initialization of the TSF through a secure process including the verification of the authenticity and integrity of the TEE firmware.
O.INSTANCE_TIME	The following requirement fulfills the objective: **FPT_STM.1/Instance time** enforces the reliability of TA instance time.
O.OPERATION	The following requirements contribute to fulfill the objective: **FAU_ARP.1** states the TEE responses to potential security violations **FDP_SDI.2** enforces the monitoring of consistency and authenticity of TEE data and TA, and it states the behavior in case of failure **FIA_ATD.1**, FIA_UID.2 and FIA_USB.1 ensure that actions are performed by identified users **FMT_SMR.1** states the two operational roles enforced by the TEE **FPT_FLS.1** states that abnormal operations have to lead to a secure state FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage and FMT_SMF.1 state the policy for controlling access to TA storage **FDP_IFC.2/Runtime** and **FDP_IFF.1/Runtime** state the policy for controlling access to TA and TEE execution spaces FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys and FMT_SMF.1 state the key access policy.
O.RNG	The requirement **FCS_RNG.1** directly fulfills the objective**.**
O.RUNTIME_CONFIDENTIALITY	The following requirements contribute to fulfill the objective: **FDP_IFC.2/Runtime** and **FDP_IFF.1/Runtime** ensure read access to authorized entities only **FDP_ITT.1/Runtime** and **FPT_ITT.1/Runtime** ensure protection against disclosure of TEE and TA data that is transferred between resources **FDP_RIP.1/Runtime** states resource clean up policy.
O.RUNTIME_INTEGRITY	The following requirements contribute to 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_ITT.1/Runtime** and **FPT_ITT.1/Runtime** ensure protection against modification of TEE and TA data that is transferred between resources **FDP_SDI.2** monitors the authenticity and consistency of TEE code, the TEE runtime data, the TA code and the TA data and keys and states the response upon failure.

Security Objective	Rationale
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, if applicable **FDP_SDI.2** monitors the authenticity and consistency of TEE persistent data and states the response upon failure **FPT_ITT.1/Runtime** enforces secure transmission and storage of TEE persistent data.
O.TEE_ISOLATION	The following requirements contribute to fulfill the objective: **FDP_IFC.2/Runtime** and **FDP_IFF.1/Runtime** state the policy for controlling access to TEE execution space.
O.TRUSTED_STORAGE	The following requirements contribute to fulfill the objective: **FCS_COP.1** states the cryptography used to protect integrity and confidentiality of the TA data in external memory, if applicable FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage and FMT_SMF.1 state Storage state the policy for accessing TA trusted storage and protecting the confidentiality of data **FDP_SDI.2** enforces the consistency and authenticity of the trusted storage **FPT_INI.1** enforces the integrity of TEE identification and storage root of trust, and it states the behavior in case of failure **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.

6.5.2 SFR Dependency Analysis

Table 6-4 SFRs Dependencies

SFRs	CC Dependencies	Satisfied Dependencies
FIA_ATD.1	No Dependencies	
FIA_UID.2	No Dependencies	
FIA_USB.1	(FIA_ATD.1)	FIA_ATD.1
FMT_SMR.1	(FIA_UID.1)	FIA_UID.2
FDP_IFC.2/Runtime	(FDP_IFF.1)	FDP_IFF.1/Runtime
FDP_IFF.1/Runtime	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/Runtime
FDP_ITT.1/Runtime	(FDP_ACC.1 or FDP_IFC.1)	FDP_IFC.2/Runtime
FDP_RIP.1/Runtime	No Dependencies	
FPT_ITT.1/Runtime	No Dependencies	
FCS_COP.1	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	
FDP_ACC.1/TA_keys	(FDP_ACF.1)	FDP_ACF.1/TA_keys
FDP_ACF.1/TA_keys	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/TA_keys, FMT_MSA.3/TA_keys
FMT_MSA.1/TA_keys	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FDP_ACC.1/TA_keys, FMT_SMF.1
FMT_MSA.3/TA_keys	(FMT_MSA.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_MSA.1/TA_keys
FAU_ARP.1	(FAU_SAA.1)	
FDP_SDI.2	No Dependencies	
FPT_FLS.1	No Dependencies	
FPT_INI.1	No Dependencies	
FMT_SMF.1	No Dependencies	
FPT_TEE.1	No Dependencies	
FAU_SAR.1	No Dependencies	
FAU_STG.1	No Dependencies	
FPT_STM.1/Instance time	No Dependencies	

SFRs	CC Dependencies	Satisfied Dependencies
FCS_RNG.1	No Dependencies	
FDP_ACC.1/Trusted Storage	(FDP_ACF.1)	FDP_ACF.1/Trusted Storage
FDP_ACF.1/Trusted Storage	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/Trusted Storage, FMT_MSA.3/Trusted storage
FDP_ROL.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted storage
FMT_MSA.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_SMF.1, FDP_ACC.1/Trusted Storage
FMT_MSA.3/Trusted Storage	(FMT_MSA.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_MSA.1/Trusted Storage
FDP_ITT.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted storage

6.5.3 SAR Rationale

Table 6-5 SARs Dependencies

Requirements	CC Dependencies	Satisfied Dependencies
ADV_ARC.1	(ADV_FSP.1) and (ADV_TDS.1)	ADV_FSP.2, ADV_TDS.1
ADV_FSP.2	(ADV_TDS.1)	ADV_TDS.1
ADV_TDS.1	(ADV_FSP.2)	ADV_FSP.2
AGD_OPE.1	(ADV_FSP.1)	ADV_FSP.2
AGD_PRE.1	No Dependencies	
ALC_CMC.2	(ALC_CMS.1)	ALC_CMS.2
ALC_CMS.2	No Dependencies	
ALC_DEL.1	No Dependencies	
ASE_CCL.1	(ASE_ECD.1) and (ASE_INT.1) and (ASE_REQ.1)	ASE_ECD.1, ASE_INT.1, ASE_REQ.2
ASE_ECD.1	No Dependencies	

Requirements	CC Dependencies	Satisfied Dependencies
ASE_INT.1	No Dependencies	
ASE_OBJ.2	(ASE_SPD.1)	ASE_SPD.1
ASE_REQ.2	(ASE_ECD.1) and (ASE_OBJ.2)	ASE_ECD.1, ASE_OBJ.2
ASE_SPD.1	No Dependencies	
ASE_TSS.1	(ADV_FSP.1) and (ASE_INT.1) and (ASE_REQ.1)	ADV_FSP.2, ASE_INT.1, ASE_REQ.2
ATE_COV.1	(ADV_FSP.2) and (ATE_FUN.1)	ADV_FSP.2, ATE_FUN.1
ATE_FUN.1	(ATE_COV.1)	ATE_COV.1
ATE_IND.2	(ADV_FSP.2) and (AGD_OPE.1) and (AGD_PRE.1) and (ATE_COV.1) and (ATE_FUN.1)	ADV_FSP.2, AGD_OPE.1, AGD_PRE.1, ATE_COV.1, ATE_FUN.1
AVA_VAN.2	(ADV_ARC.1) and (ADV_FSP.2) and (ADV_TDS.1) and (AGD_OPE.1) and (AGD_PRE.1)	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1
AVA_TEE.2	(ADV_ARC.1) and (ADV_FSP.2) and (ADV_TDS.1) and (AGD_OPE.1) and (AGD_PRE.1)	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1

6.5.4 SAR Dependency Analysis

The Evaluation Assurance Level 2 has been chosen to commensurate with the threat environment that is experienced by typical consumers of the TOE.

The assurance level defined in this ST consists of the predefined assurance package EAL 2 with the augmentation AVA_TEE.2 (extended SAR TEE Vulnerability analysis) in order to reach the TEE-Low attack potential.

This augmented EAL permits a developer to gain sufficient assurance from positive security engineering based on good TEE commercial development practices that are compatible with industry constraints, in particular the life cycle of TEE and TEE-enabled devices. The developer has to provide evidence of security engineering at design, testing, guidance, and configuration management and delivery levels as required by standard 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 TEE-Low attack potential. This attack potential matches the threat analysis performed on typical architectures and attack profiles in the field.

The components AVA_VAN.2 and AVA_TEE.2 are chosen together in the augmented EAL package, while they should normally be exclusive. The reason of this choice is to perform the attack quotation according to the two tables and to allow EAL2 product recognition for the schemes that do not recognize the AVA_TEE.2 component.

7 TOE Summary Specification

This section presents the TOE Security Functions (TSFs) and a mapping of security functions to Security Functional Requirements (SFRs).

The TOE performs the following security functions:

1) Protection of TSF
2) Trusted Storage
3) Cryptographic Support
4) Reliable time stamps
5) User Identification and Authentication
6) Security Audit
7) Protection of TA
8) Communication Data Protection between CA and TA
9) Security Instantiation

7.1 Protection of TSF

The TOE provides several security mechanisms to ensure self security. Detailed functions include:

- The TOE provides a system-wide hardware isolation for the iTrustee OS. The Rich OS (eg. Android) running in the normal world cannot access any data directly from secure world where the iTrustee OS is running.
- The CA in normal world can only communicate with the secure world by sending SMC call, and the iTrustee OS will determine whether to reply or not according to its secure policy.
- TAs are encrypted and signed by Huawei, and will be decrypted and the signature is checked before loading into the TOE. The TOE will stop loading TA if tampering is detected.
- The TOE will enter a secure state when some key operation error occurred, to protect the TOE's data from leaking out.

(FPT_TEE.1, FPT_FLS.1, FDP_SDI.2, FMT_SMF.1, FMT_SMR.1, FDP_ITT.1/Runtime, FPT_ITT.1/ Runtime)

7.2 Trusted Storage

The TOE provides trusted storage service though the Trusted core framework. TAs can load or store its owning files through Trusted Storage service, and any intention to access other TA's file will be prevented.

- The trusted storage service is the only task that has the permission to access the file in Trusted Storage. The TOE will reject any request to access the trusted file from other tasks.
- The trusted storage service of the TOE will receive a *trusted file access* request from a TA, if the request is illegal or the TA has no permission to access the trusted file, the request will be denied.
- The trusted storage service will encrypt the data before writing the data into the trusted file, and the encrypted key is bind to the requester TA. Different TAs use different keys. The trusted storage service will decrypt the data and send back to the reading request TA.
- If the file write operation fails, the TOE will detect the failure and rollbacks to previous state.

(FDP_ACC.1/Trusted storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FDP_ITT.1/Trusted Storage).

7.3 Cryptographic Support

The TOE provides cryptographic operations to support TOE's security functions such as CA authentication, TA signature verification, communication encryption, trust storage etc. The TOE provides software cryptographic operations.

- The TOE provides a series of cryptographic operations such as hash calculation, HMAC calculation, signature generation, encryption and decryption in accordance with the cryptographic algorithms specified in Table 1-3.
- The TOE provides TA_Keys in accordance with GP Internal API Specification.
- The TOE provides RSA 2048 and SHA256 signature verification of TEE firmware upon initialization.
- The TOE provides the random numbers generation.

(FCS_COP.1,FDP_ACC.1/TA_Keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys, FCS_RNG.1)

7.4 Reliable time stamps

The TOE get security time from the SoC, and provides unified time stamps to all the TAs. The time stamp provided by the TOE will increase monotonically from the TOE starting-up, no matter the TOE is running or sleeping.

(FPT_STM.1)

7.5 User Identification and Authentication

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

(FIA_ATD.1, FIA_UID.2, FIA_USB.1)

7.6 Security Audit

The TOE will detect potential security violation such as violation of TA data, TA code or TEE data, and generate audit log. The TOE will send the log info from TEE to REE, which can be read by human with tool.

Every audit log record of TOE contains TEE identifier which is generated on-TEE.

(FAU_ARP.1, FAU_SAR.1, FAU_STG.1)

7.7 Protection of TA

When TA is being loaded, the TSF will check the signature of it so that integrity and authenticity are ensured.

For each loaded TA, the TSF will create an isolated running environment, any reading or writing accesses from other TAs are forbidden. And TSF will monitor the authenticity and consistency of TA code, data and keys, when a failure occurs, the TSF will cleanup all those data and the TA will exit.

(FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_RIP.1/Runtime, FDP_SDI.2)

7.8 Communication Data Protection between CA and TA

The CA user must complete identity authentication before performing any further actions, and it can only communicate with the specified TA which is pre-defined in the TOE.

The TSF encrypts the key data of the communication to prevent data sniffing from the REE world, adds token info to prevent data replay attack from the REE world, and uses checksum to prevent data modification.

(FDP_IFC.2/Runtime, FDP_IFF.1/Runtime,FDP_RIP.1/Runtime)

7.9 Security Instantiation

The TEE instantiation is through a secure initialization process using assets bound to the SoC, that ensures the authenticity and contributes to the integrity of the TEE code running in the device;

Each step of the startup process contains components that are cryptographically signed to ensure integrity and that proceed only after verifying the chain of trust. The chain of trust includes the onchiprom, fastboot, bootloaders, kernel, and so on. This secure boot chain helps ensure that TOE's instantiation are not tampered with.

(FPT_INI.1,FCS_COP.1)

8 Acronyms

Table 8-1 Acronyms

Acronym	Meaning
ST	Security Target
PP	Protection Profile
CC	Common Criteria
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFi	TSF Interface
IT	Information Technology
OSP	Organisational Security Policies
EAL	Evaluation Assurance Level
TSS	TOE Summary Specification

9 Glossary of Terms

Table 9-1 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
Target Of Evaluation	Set of software, firmware and/or hardware possibly accompanied by guidance

10 Document References

The following table shows the documents referenced in this Security Target

Table 10-1 Document References

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
CEM31R5	Common Criteria Evaluation methodology, Version 3.1, Revision 5
WP	The Trusted Execution Environment: Delivering Enhanced Security at a Lower Cost to the Mobile Market, GlobalPlatform White paper,Feb 2011
TEE PP	GlobalPlatform Device Committee TEE Protection Profile Version 1.2.1

