GLOBALPLATFORM

THE STANDARD FOR MANAGING APPLICATIONS ON SECURE CHIP TECHNOLOGY

GlobalPlatform Device Committee

TEE Protection Profile

Version 1.2.1

Public Release

November 2016

Document Reference: GPD_SPE_021

This page intentionally left blank.

Contents

1 INTRODUCTION 7
1.1 AUDIENCE 8
1.2 IPR DISCLAIMER 8
1.3 REFERENCES 8
1.4 TERMINOLOGY AND DEFINITIONS 10
1.5 ABBREVIATIONS AND NOTATIONS 13
1.6 REVISION HISTORY 15
2 TOE OVERVIEW 16
2.1 TOE TYPE 16
2.2 TOE DESCRIPTION 16
2.2.1 SOFTWARE ARCHITECTURE OF A TEE-ENABLED DEVICE 16
2.2.2 HARDWARE ARCHITECTURE OF A TEE-ENABLED DEVICE 17
2.3 USAGE AND MAJOR SECURITY FEATURES OF THE TOE 20
2.3.1 TEE SECURITY FUNCTIONALITY 21
2.3.2 TOE USAGE 21
2.3.3 TEE TIME AND ROLLBACK PP-MODULE 22
2.3.4 TEE DEBUG PP-MODULE 22
2.4 AVAILABLE NON-TOE HARDWARE/SOFTWARE/FIRMWARE 23
2.5 REFERENCE DEVICE LIFE CYCLE 23
3 CONFORMANCE CLAIMS AND CONSISTENCY RATIONALE 26
3.1 CONFORMANCE CLAIM TO CC 26
3.2 CONFORMANCE CLAIM TO A PACKAGE 26
3.3 CONFORMANCE CLAIM OF THE PP 26
3.4 CONFORMANCE CLAIM TO THE PP 26
3.5 CONSISTENCY RATIONALE FOR THE PP-MODULES 26
3.5.1 TEE TIME AND ROLLBACK PP-MODULE 26
3.5.2 TEE DEBUG PP-MODULE 27
4 SECURITY PROBLEM DEFINITION 28
4.1 ASSETS 28
4.1.1 TEE BASE-PP 28
4.1.2 TEE TIME AND ROLLBACK PP-MODULE 29
4.1.3 TEE DEBUG PP-MODULE 30
4.2 USERS / SUBJECTS 30
4.2.1 TEE BASE-PP 30
4.2.2 TEE DEBUG PP-MODULE 31
4.3 THREATS 31
4.3.1 TEE BASE-PP 32
4.3.2 TEE TIME AND ROLLBACK PP-MODULE 35
4.3.3 TEE DEBUG PP-MODULE 35
4.4 ORGANIZATIONAL SECURITY POLICIES 36
4.4.1 TEE BASE-PP 36
4.4.2 TEE TIME AND ROLLBACK PP-MODULE 36
4.4.3 TEE DEBUG PP-MODULE 36
4.5 ASSUMPTIONS 36
4.5.1 TEE BASE-PP 36
4.5.2 TEE TIME AND ROLLBACK PP-MODULE 37
4.5.3 TEE DEBUG PP-MODULE 37

5 SECURITY OBJECTIVES 38
5.1 SECURITY OBJECTIVES FOR THE TOE 38
5.1.1 TEE BASE-PP 38
5.1.2 TEE TIME AND ROLLBACK PP-MODULE 41
5.1.3 TEE DEBUG PP-MODULE 41
5.2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT 41
5.2.1 TEE BASE-PP 42
5.2.2 TEE TIME AND ROLLBACK PP-MODULE 42
5.2.3 TEE DEBUG PP-MODULE 43
5.3 SECURITY OBJECTIVES RATIONALE 43
5.3.1 THREATS 43
5.3.2 ORGANIZATIONAL SECURITY POLICIES 46
5.3.3 ASSUMPTIONS 46
5.3.4 SPD AND SECURITY OBJECTIVES 46
6 EXTENDED REQUIREMENTS 52
6.1 EXTENDED FAMILIES 52
6.1.1 EXTENDED FAMILY FCS_RNG - GENERATION OF RANDOM NUMBERS 52
6.1.2 EXTENDED FAMILY FPT_INI - TSF INITIALISATION 53
6.1.3 EXTENDED FAMILY AVA_TEE - VULNERABILITY ANALYSIS OF TEE 54
7 SECURITY REQUIREMENTS 56
7.1 SECURITY FUNCTIONAL REQUIREMENTS 56
7.1.1 TEE BASE-PP 56
7.1.2 TEE TIME AND ROLLBACK PP-MODULE 71
7.1.3 TEE DEBUG PP-MODULE 73
7.2 SECURITY ASSURANCE REQUIREMENTS 76
7.3 SECURITY REQUIREMENTS RATIONALE 76
7.3.1 OBJECTIVES 76
7.3.2 RATIONALE TABLES OF SECURITY OBJECTIVES AND SFRS 79
7.3.3 DEPENDENCIES 83
7.3.4 RATIONALE FOR THE SECURITY ASSURANCE REQUIREMENTS 87
A. APPLICATION OF ATTACK POTENTIAL TO TEE 89
A.1 ATTACK QUOTATION GRID 89
A.2 ATTACKERS' EXPLOITATION PROFILES 96
A.2.1 EXPLOITATION PROFILE 1 (REMOTE ATTACKER) 98
A.2.2 EXPLOITATION PROFILE 2 (LOCAL LAYMAN ATTACKER) 98
A.2.3 EXPLOITATION PROFILE 3 (LOCAL PROFICIENT ATTACKER) 98
A.2.4 EXPLOITATION PROFILE 4 (LOCAL PROFICIENT ATTACKER WITH EQUIPMENT) 98
A.3 EXAMPLES OF ATTACK PATHS 99
A.3.1 HARDWARE-BASED ATTACK PATHS 99
A.3.1.1 SIDE CHANNEL ANALYSIS ATTACK 99
A.3.1.2 FAULT INJECTION ATTACK 99
A.3.1.3 EXTERNAL DRAM PROBING 100
A.3.1.4 UNPROTECTED DEBUG INTERFACE 100
A.3.2 SOFTWARE-BASED ATTACK PATHS 100
A.3.2.1 CACHE ATTACK ON CRYPTO 100
A.3.2.2 FUZZING ON THE CLIENT API OR THE TEE DRIVER 101
A.3.2.3 BREACH OF MEMORY ISOLATION 101
A.3.2.4 CERTIFICATE PARSING ERROR 101
A.3.2.5 USE OF APIS/PROTOCOLS WITH KNOWN VULNERABILITIES OR DIAGNOSTIC APIS 102

Figures

FIGURE 2-1: TEE OVERALL SOFTWARE ARCHITECTURE 17
FIGURE 2-2: SEPARATED TRUSTED AND UN-TRUSTED RESOURCES. 19
FIGURE 2-3: EXAMPLES OF TEE REALIZATIONS 20
FIGURE 2-4 –LIFE CYCLE OF TEE-ENABLED DEVICE 24

Tables

TABLE 1: NORMATIVE REFERENCES 9
TABLE 2: TERMINOLOGY AND DEFINITIONS 13
TABLE 3: ABBREVIATIONS AND NOTATIONS 15
TABLE 4: ACTORS IN THE DEVICE LIFE CYCLE 25
TABLE 5 THREATS AND SECURITY OBJECTIVES - COVERAGE 48
TABLE 6 SECURITY OBJECTIVES AND THREATS - COVERAGE 50
TABLE 7 OSPS AND SECURITY OBJECTIVES - COVERAGE 50
TABLE 8 SECURITY OBJECTIVES AND OSPS - COVERAGE 51
TABLE 9 ASSUMPTIONS AND SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT - COVERAGE 51
TABLE 10 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT AND ASSUMPTIONS - COVERAGE 51
TABLE 11 SECURITY OBJECTIVES AND SFRS - COVERAGE 81
TABLE 12 SFRS AND SECURITY OBJECTIVES 83
TABLE 13 SFRS DEPENDENCIES 85
TABLE 14 SARS DEPENDENCIES 87
TABLE 15: TEE ATTACK POTENTIAL 91
TABLE 16: EQUIPMENT RATING TABLE FOR TEE ATTACKS 94
TABLE 17: RATING OF TEE RESISTANCE 95
TABLE 18: RATINGS OF EXPLOITATION PROFILES 1 TO 4 97

1 Introduction

Title:	TEE Protection Profile (base PP and optional TEE Time and Rollback PP-module and TEE Debug PP-module)
Identifications:	GPD_SPE_021 (PP-configuration composed of the base Protection Profile only)
	GPD_SPE_021+Time (PP-configuration composed of the base Protection Profile and the TEE Time and Rollback PP-module)
	GPD_SPE_021+Debug (PP-configuration composed of the base Protection Profile and the TEE Debug PP-module)
	GPD_SPE_021+Time&Debug (PP-configuration composed of the base Protection Profile and the TEE Time and Rollback and TEE Debug PP-modules)
Editor:	Trusted Labs
Date:	November 2016
Version:	1.2.1
Sponsor:	GlobalPlatform
CC Version:	3.1 Revision 4

This Protection Profile (PP) has been developed by the Security Working Group of the GlobalPlatform Device Committee. It constitutes the reference for the Common Criteria (CC) evaluation of GlobalPlatform Trusted Execution Environment (TEE), which aim at enabling mobile security services such as content protection, rights management, corporate policies, payment, etc.

The TEEs in the scope of this PP implement the core functionalities defined in GlobalPlatform TEE Internal API Specification [IAPI]. This PP relies on the Common Criteria Modular Protection Profile methodology [PP-MOD] to define a ‘base-PP’ with the minimum TEE security requirements and optional ‘PP-modules’ that apply to those TEEs that implement full rollback protection and persistent monotonic time and those TEEs that allow access to debug features. These ‘PP-modules’ can be used with the ‘base-PP’ to compose a ‘PP-configuration’. This document supports all the combinations of the ‘base-PP’ with the two ‘PP-modules’ introduced above.

This Protection Profile claims conformance with EAL 2 package augmented with an extended security assurance requirement called AVA_TEE.2. This extended SAR aims at raising the attack potential over the standard Basic attack potential defined for AVA_VAN.2 in [CC Part 3] and [CEM]. The evaluation methodology, including the specific attack potential quotation table and a representative set of TEE attacks, is defined in Annex A. This extended SAR can only be used for product evaluation if it is recognized by the Certification Body that monitors the product evaluation otherwise the conformance claim is limited to EAL 2.

1.1 Audience

This document is dedicated to all actors in the TEE value chain: TEE developers, integrators (in particular handset makers), service providers (TA developers), as well as ITSEFs, certification bodies and Common Criteria certificate consumers.

1.2 IPR Disclaimer

GlobalPlatform draws attention to the fact that claims that compliance with this specification may involve the use of a patent or other intellectual property right (collectively, "IPR") concerning this specification may be published at https://www.globalplatform.org/specificationsipdisclaimers.asp. GlobalPlatform takes no position concerning the evidence, validity, and scope of these IPR claims.

1.3 References

Standard / Specification	Description	Ref.
CC Part 1	Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1, revision 4, September 2012. CCMB-2012-09-001.	[CC1]
CC Part 2	Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1, revision 4, September 2012. CCMB-2012-09-002.	[CC2]
CC Part 3	Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements. Version 3.1, revision 4, September 2012. CCMB-2012-09-003.	[CC3]
CEM	Common Methodology for Information Technology Security Evaluation, Evaluation Methodology. Version 3.1, revision 4, September 2012. CCMB-2012-09-004.	[CEM]
CEM addenda	CC and CEM addenda, Modular PP, Version 1.0, March 2014. CCDB-2014-03-001	[PP-MOD]
CC Supporting Document	Application of Attack Potential to Smartcards. Version 2.9 January 2013. Joint Interpretation Library.	[APSC]
OMTP ATE TR1	Open Mobile Terminal Platform Advanced Trusted Environment OMTP TR1 v1.1	[OMTP-TR1]
OMTP Security Threats	OMTP Security Threats on Embedded Consumer Devices v1.1	[OMTP-ST]
TEE White Paper	The Trusted Execution Environment: Delivering Enhanced Security at a Lower Cost to the Mobile Market, GlobalPlatform White paper, Feb 2011	[WP]
GPD_SPE_009	TEE System Architecture, GlobalPlatform (Last applicable version)	[SA]
GPD_SPE_010	TEE Internal API Specification, GlobalPlatform (Last applicable version)	[IAPI]

Standard / Specification	Description	Ref.
GPD_SPE_007	TEE Client API Specification, GlobalPlatform (Last applicable version)	[CAPI]
FIPS Publication	ADVANCED ENCRYPTION STANDARD (AES). FIPS PUB 197. November 2011.	[AES]
FIPS Publication	DATA ENCRYPTION STANDARD (DES). FIPS PUB 46-3. October 1999.	[DES]
RSA Laboratories Publication	RSA Cryptographic Standard. PCKS#1 v2.2. October 2012	[RSA]
FIPS Publication	SECURE HASH STANDARD. FIPS PUB 180-4. March 2012	[SHA]
IEEE Standard	IEEE 1149.1-2001 Standard Test Access Port and Boundary-Scan Architecture http://standards.ieee.org/reading/ieee/std_public/description/testtech/1149.1-2001_desc.html	[JTAG]
RFC2119	Key words for use in RFCs to Indicate Requirement Levels	[RFC2119]

Table 1: Normative References

1.4 Terminology and Definitions

Throughout this document, normative requirements are highlighted by use of capitalized key words as described below.

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119]:

- MUST - This word, or the terms “REQUIRED” or “SHALL”, mean that the definition is an absolute requirement of the specification
- MUST NOT - This phrase, or the phrase “SHALL NOT”, mean that the definition is an absolute prohibition of the specification
- SHOULD - This word, or the adjective “RECOMMENDED”, mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course
- SHOULD NOT - This phrase, or the phrase “NOT RECOMMENDED” mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

MAY - This word, or the adjective “OPTIONAL”, mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)

Table 1 defines the expressions used within this Protection Profile that use an upper case first letter in each word of the expression. Expressions within this document that use a lower case first letter in each word take the common sense meaning. CC terminology, defined in [CC1] §4, is not listed here.

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

Term	Definition
Rich Execution Environment (REE)	An environment that is provided and governed by a Rich OS, potentially in conjunction with other supporting operating systems and hypervisors; it is outside of the TEE. This environment and applications running on it are considered un-trusted. Contrast *Trusted Execution Environment*.
Rich OS	Typically an OS providing a much wider variety of features than that of the OS running inside the TEE. It may be very open in its ability to accept applications. It will have been developed with functionality and performance as key goals, rather than security. Due to the size and needs of the Rich OS it will run in an execution environment that may be larger than the TEE hardware (often called an REE – Rich Execution Environment) with much lower physical security boundaries. From the TEE viewpoint, everything in the REE has to be considered un-trusted, though from the Rich OS point of view there may be internal trust structures. Contrast *Trusted OS*.
Root of Trust (RoT)	Generally the smallest distinguishable set of hardware, firmware, and/or software that must be inherently trusted and which is closely tied to the logic and environment on which it performs its trusted actions.
System-on-Chip (SoC)	An electronic system all of whose components are included in a single integrated circuit.
TA instance time / TA persistent time	Time value available to a Trusted Application through the TEE Internal API. The API offers two types of time values: System Time, which exists only during runtime, and Persistent time, which persists over resets. System Time must be monotonic for a given TA instance, and the returned value is called “TA instance time”. Persistent time depends only on the TA but not on a particular instance, it must be monotonic even across power cycles. Its monotonicity across power cycles is related to the Time and Rollback optional PP-module.
TEE Client API	The software interface used by clients running in the REE to communicate with the TEE and with the Trusted Applications executed by the TEE.
TEE Communication Agent	A TEE Trusted OS driver that enables communication between REE and TEE. Contrast *REE Communication Agent*.
TEE Internal API	The software interface exposing TEE functionality to Trusted Applications.
TEE Service Library	A software library that includes all security related drivers.
Trusted Application (TA)	An application running inside the Trusted Execution Environment that exports security related functionality to Client Applications outside of the TEE. Contrast *Client Application*.

Term	Definition
Trusted Execution Environment (TEE)	An execution environment that runs alongside but isolated from an REE. A TEE has security capabilities and meets certain security-related requirements: It protects TEE assets from general software attacks, defines rigid safeguards as to data and functions that a program can access, and resists a set of defined threats. There are multiple technologies that can be used to implement a TEE, and the level of security achieved varies accordingly. For more information, see OMTP ATE TR1 [OMTP-TR1]. Contrast *Rich Execution Environment.*
Trusted OS	The operating system running in the TEE. It has been designed primarily to enable the TEE using security-based design techniques. It provides the GlobalPlatform TEE Internal API to Trusted Applications and a proprietary method to enable the GlobalPlatform TEE Client API software interface from other EE. Contrast *Rich OS*.
Trusted Storage	In GlobalPlatform TEE documents, *trusted storage* indicates storage that is protected to at least the robustness level defined for OMTP Secure Storage (in section 5 of [OMTP-TR1]). It is protected either by the hardware of the TEE, or cryptographically by keys held in the TEE. If keys are used they are at least of the strength used to instantiate the TEE. A GlobalPlatform TEE Trusted Storage is not considered hardware tamper resistant to the levels achieved by Secure Elements.

Table 2: Terminology and Definitions

1.5 Abbreviations and Notations

Table 3 defines the abbreviations used within this Protection Profile.

Abbreviation	Meaning
AES	Advanced Encryption Standard (defined in [AES])
API	Application Programming Interface
CA	Client Application
CC	Common Criteria (defined in [CC1], [CC2], [CC3])
CEM	Common Evaluation Methodology (defined in [CEM])
CM	Configuration Management (defined in [CC1])
DES	Data Encryption Standard (defined in [DES])
DRM	Digital Rights Management
EAL	Evaluation Assurance Level (defined in [CC1])

Abbreviation	Meaning
EE	Execution Environment
ID	IDentifier
FIFO	First In, First Out
HD	High-Definition
HDMI	High-Definition Multimedia Interface
IPsec	Internet Protocol security
JTAG	Joint Test Action Group (defined in [JTAG])
MAC	Message Authentication Code
NA	Not Applicable
NFC	Near Field Communication
OMTP	Open Mobile Terminal Platform
OS	Operating System
OSP	Organisational Security Policy (defined in [CC1])
OTP	One-Time Password
PCB	Printed Circuit Board
PP	Protection Profile (defined in [CC1])
RAM	Random Access Memory
REE	Rich Execution Environment
RFC	Request For Comments; may denote a memorandum published by the IETF
ROM	Read Only Memory
RSA	Rivest / Shamir / Adleman asymmetric algorithm (defined in [RSA])
SAR	Security Assurance Requirement (defined in [CC1])
SFP	Security Function Policy (defined in [CC1])
SFR	Security Functional Requirement (defined in [CC1])
SHA	Secure Hash Algorithm (defined in [SHA])
SoC	System-on-Chip
SPD	Security Problem Definition (defined in [CC1])
SSL	Secure Sockets Layer
ST	Security Target (defined in [CC1])
TA	Trusted Application
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TOE	Target of Evaluation (defined in [CC1])

Abbreviation	Meaning
TSF	TOE Security Functionality (defined in [CC1])
TSFI	TSF Interface (defined in [CC1])
USB	Universal Serial Bus
VPN	Virtual Private Network

Table 3: Abbreviations and Notations

1.6 Revision history

Date	Version	Author	Description
09/08/2013	0.5.3 / 1.0	C. Lavatelli, Trusted Labs	Update following Public Review
29/09/2014	1.1	G. Dufay, Trusted Labs	Update during PP evaluation
18/11/2014	1.2	G. Dufay, Trusted Labs	Typos and minor clarifications
25/11/2016	1.2.1	C. Lavatelli, GlobalPlatform SES	Update of Annex A (attack quotation table and examples)

2 TOE Overview

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

2.1 TOE Type

The TOE type is the 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]). However, this Protection Profile does not require full functional compliance with GlobalPlatform TEE APIs specifications.

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.

The TOE comprises:

- Any hardware, firmware and software used to provide the TEE security functionality
- The guidance for the secure usage of the TEE after delivery.

The TOE does not comprise:

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

In the following, TOE and TEE are used interchangeably.

2.2 TOE Description

2.2.1 Software Architecture of a TEE-enabled Device

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

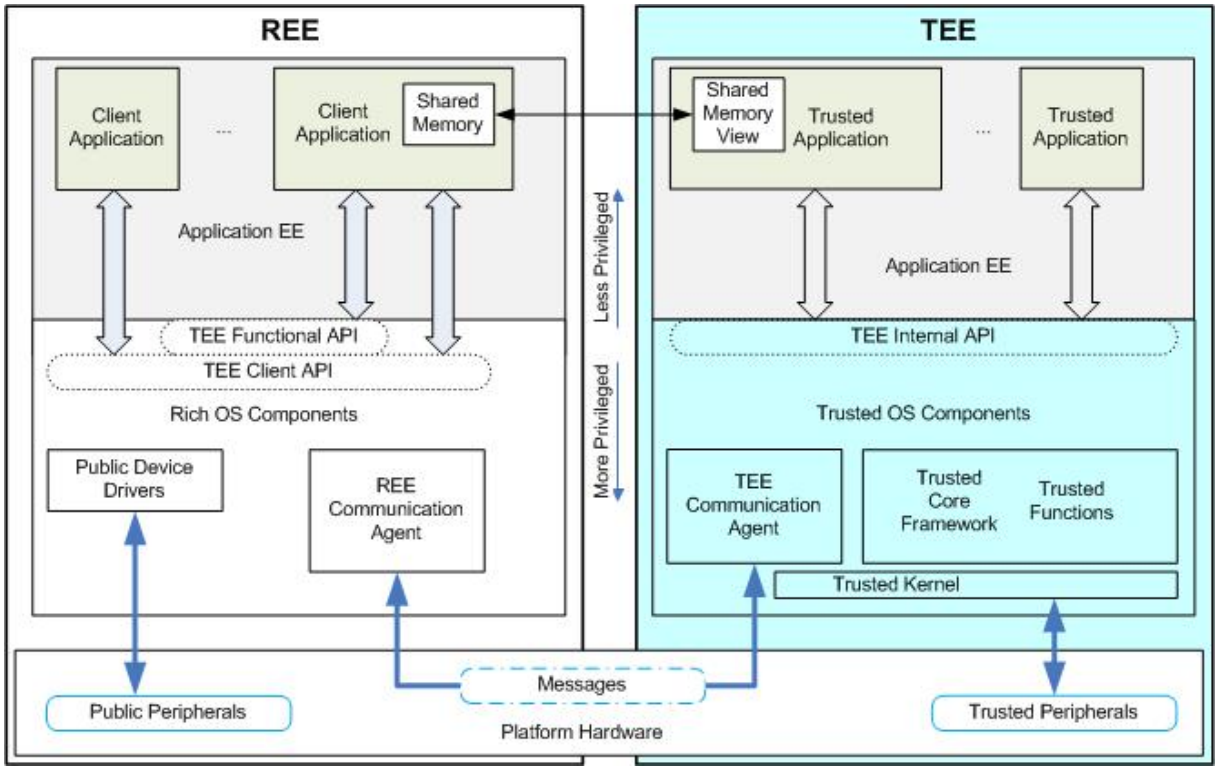

Figure 2-1: TEE Overall Software Architecture

The TEE software architecture identifies two distinct classes of components:

- The Trusted Applications that run on the TEE and use the TEE Internal API
- The Trusted OS 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 TEE software external interface comprises the TEE Internal API (used by the Trusted Applications) and the TEE Communication Agent protocol (used by the REE).

The communication protocol between the REE and the TEE, used below the TEE Client API level, is implementation-dependent, and therefore this Protection Profile does not mandate any particular such protocol. The security targets conformant to this PP shall describe all software interfaces used for communication with the TEE from the REE.

2.2.2 Hardware Architecture of a TEE-enabled Device

The TEE is embedded in a device platform including:

- Hardware processing unit(s)

- Hardware resources such as
  - Physical volatile memory
  - Physical non-volatile memory
  - Peripherals, like keyboard and display
  - Cryptographic accelerators
  - Secure clock
  - Secure element
- A set of connections between the processing unit(s) and the hardware resources

Schematically, a TEE-enabled device is structured in four layers:

- The *die layer*, System-on-Chip (SoC), which contains processor(s) and resources such as memories, crypto-accelerators, peripherals (e.g. JTAG, USB, serial, HDMI), etc.
- The *package layer*, which embeds the SoC and contains further resources, e.g. non-volatile and volatile memories, pins or buses. Resources inside the same package layer are connected using buses that are not externally accessible. External buses in the specification are outside the package layer. “3D” die stacking techniques may be used to place more facilities inside the package that may not be in the die layer.
- The PCB layer, which contains SoC, package, non-volatile and volatile memories, wireless and contactless interface chips, security modules and other resources.
- The *user layer*, which contains user interfaces to the package, such as the touch screen or keyboard, and may contain other resources.

The TEE is typically implemented in the die and package layers of one package but it may be instantiated in a number of separate packages using cryptographic linking (secure channels) between TEE components. The TEE hardware external interface stands for the package input and output interfaces, which provide access to the package resources and indirectly to the SoC internals, both from the user layer and from the SoC itself. This PP considers the package internals as a black-box.

Nevertheless, the physical boundary of the TEE is implementation-dependent. Furthermore, the set of “trusted” resources used to realize the security functionality, which is controlled by the TEE, can change dynamically. For instance, some communication resources such as the keyboard may sometimes be within the TEE boundaries if the TEE enforces exclusive access to these resources. From a logical point of view, the “trusted” resources used by the TEE are separated from the “un-trusted” resources used by the REE. That is, the TEE and the REE coexist in the device but isolated from each other, as shown in Figure 2-2.

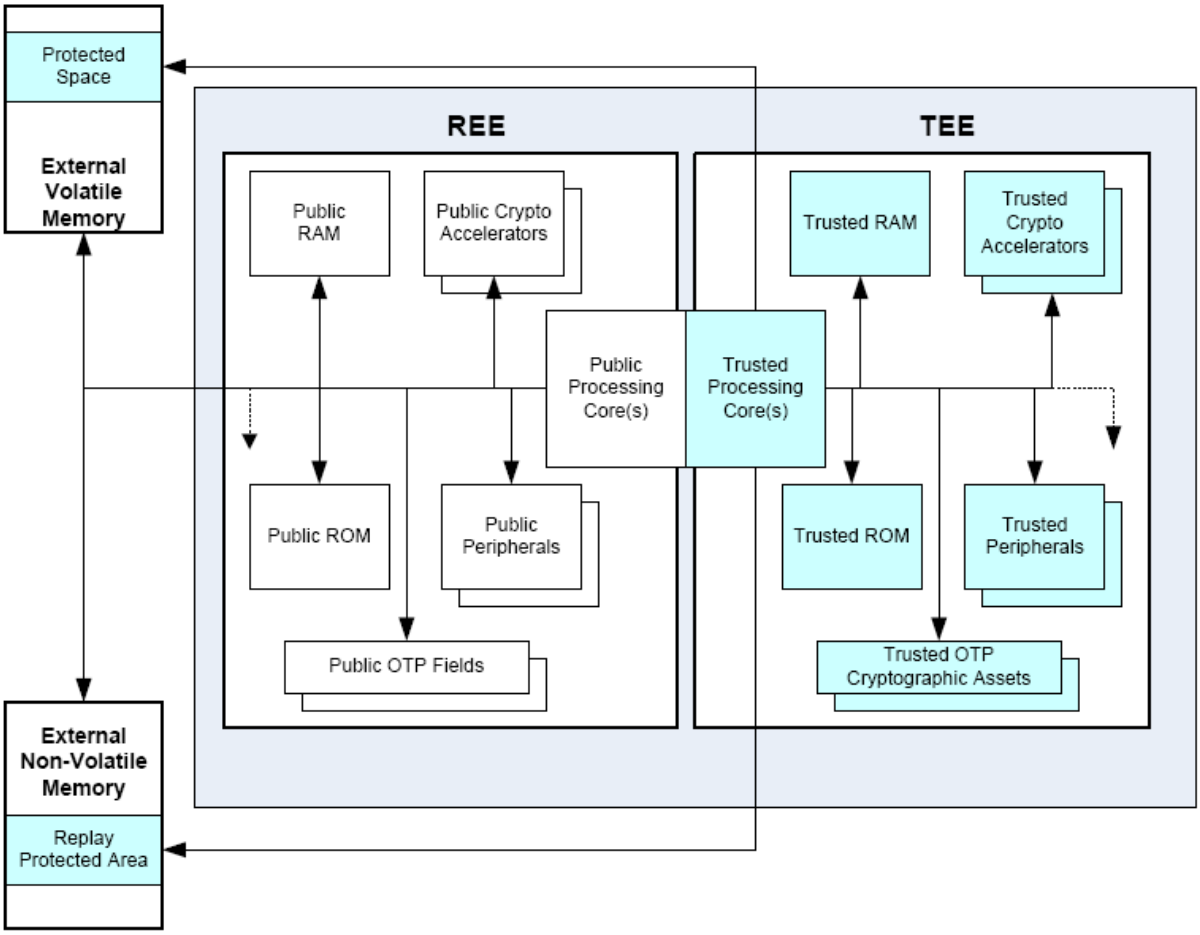

Figure 2-2: Separated Trusted and Un-trusted Resources.

In practice, there are several ways to architect a TEE within a device and to isolate it from the REE. Figure 2-3 illustrates three possible realizations, with different resource-sharing policies between the TEE and the REE. Indeed, the TEE and the REE can share device resources provided the TEE controls access to them.

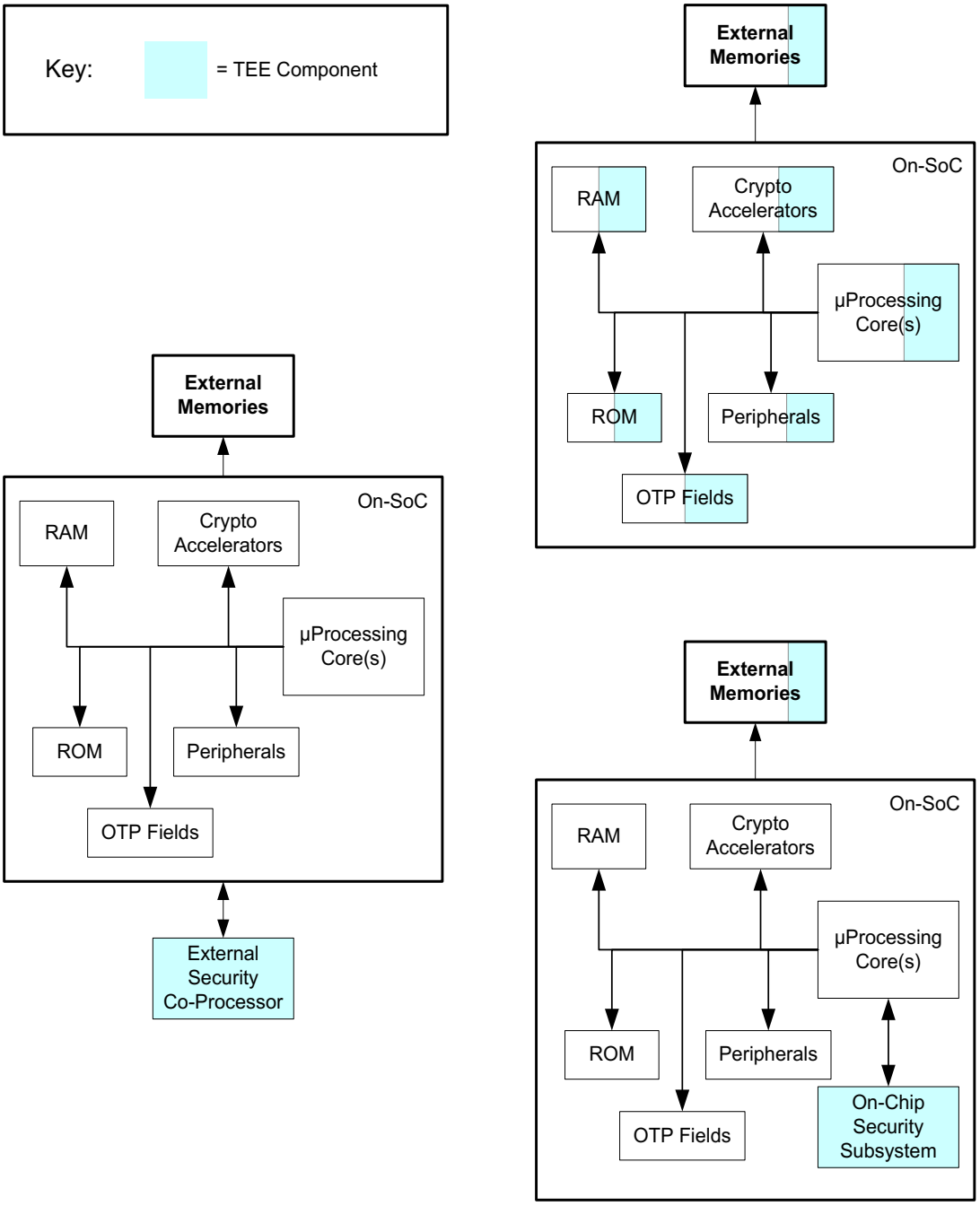

Figure 2-3: Examples of TEE Realizations

This Protection Profile does not mandate any particular hardware architecture, resource set or isolation mechanisms from the REE. The security targets conformant to this PP shall describe the physical layout and precisely define the physical boundaries of the TEE and the hardware external interface.

2.3 Usage and Major Security Features of the TOE

The purpose of the TEE is to host and execute Trusted Applications securely, enforcing their mutual isolation and isolation from other execution environments, and ensuring integrity and confidentially of the assets managed by the TEE.

The following sections define the TEE security functionality and the TEE intended usage.

2.3.1 TEE Security Functionality

The TEE security functionality in the end-user phase (cf. section 2.5) which is in the scope of the evaluation consists of:

- TEE instantiation through a secure initialization process using assets bound to the SoC, that ensures the authenticity and contributes to the integrity of the TEE code running in the device
- Isolation of the TEE services, the TEE resources involved and all the Trusted Applications from the REE
- Isolation between Trusted Applications and isolation of the TEE from Trusted Applications
- Protected communication interface between CAs and TAs within the TEE, including communication endpoints in the TEE
- Trusted storage of TA and TEE data and keys, ensuring consistency (cf. section 1.4), confidentiality, atomicity and binding to the TEE
- Random Number Generator
- Cryptographic API including:
    - Generation and derivation of keys and key pairs
    - Support for cryptographic algorithms such as SHA-256, AES 128/256, T-DES, RSA 2048, etc. (this list is for example only, see the Application Note below)
- TA instantiation that ensures the authenticity and contributes to the integrity of the TA code
- Monotonic TA instance time
- Correct execution of TA services
- TEE firmware integrity verification
- Prevention of downgrade of TEE firmware.

The TEE security functionality defines the logical boundary of the TOE. The interfaces of this boundary are the Software External Interface and the Hardware External Interface, introduced in sections 2.2.1 and 2.2.2 respectively.

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

Application Note: Security Targets conformant to this PP shall complete the descriptions of the security functionality with the characteristics of the actual TOE, including any TA management functionalities if applicable (on top of verification of TA authenticity prior to execution), and the complete list of cryptographic algorithms supported by the product.

2.3.2 TOE Usage

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

- Corporate services: Enterprise devices that enable push e-mail access and office applications give employees a flexibility that requires a secure and fast link to their workplace applications through Virtual Private Networking (VPN), secure storage of their data, and remote management of the device by the IT department.

- Content management: Today's devices offer HD video playback and streaming, mobile TV broadcast reception, and console-quality 3-D games. This functionality often requires content protection, through Digital Rights Management (DRM) or Conditional Access.

- Personal data protection: Devices store increasing amounts of personal information (such as contacts, messages, photos and video clips) and even sensitive data (credentials, passwords, health data, etc.). Secure storage means are required to prevent exposure of this information in the event of loss, theft, or any other adverse event, such as a malware.

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

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

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

2.3.3 TEE Time and Rollback PP-Module

The TEE Time and Rollback PP-module addresses the following security functionality, which complements the core functionality defined in section 2.3.1:

1. Monotonic TA persistent time
2. Integrity verification of TA trusted storage (data and keys)
3. Integrity verification of TA code and configuration data

Notice that monotonic persistent time allows a service to be delivered after a power cycle without any remote help. For connected services that can get an updated time at startup, monotonic instance time may be sufficient.

2.3.4 TEE Debug PP-Module

The TEE Debug PP-module addresses the access to TEE Debug functionality for the TEE Debug Administrator, as in the core Configuration this functionality is not supported.

2.4 Available Non-TOE Hardware/Software/Firmware

The TOE may require some non-TOE Hardware, Software or Firmware in order to operate, such as non-volatile memory. However, the TOE must be realized in a way such that TOE security functionalities do not rely on proper behavior of non-TOE hardware, software or firmware.

Application note: Security Targets conformant to this PP shall complete the descriptions of the available non-TOE hardware/software/firmware with the list of non-TOE resources used by the TOE.

2.5 Reference Device Life Cycle

The device life cycle outlined here is a reference life cycle from which implementations can deviate according to development, manufacturing and assembly processes. It is split in six phases:

- Phase 1 corresponds to the design of firmware, software and hardware; it covers both TEE and additional components
- Phase 2 corresponds to the overall design of the hardware platform supporting the TEE
- Phase 3 corresponds to chipset and other hardware components manufacturing
- Phase 4 covers software preparation (e.g. linking the TEE software and other software)
- Phase 5 consists of device assembling; it includes any initialization and configuration step necessary to bring the device to a secure state prior delivery to the end-user
- Phase 6 stands for the end-usage of the device

Secure boot/firmware, including TEE initialization code, is usually installed in phase 3 though it may be upgraded later. The root of trust of the TEE storage services and the TEE unique identifier are set (injection or on-board creation) in phase 3 or 5. The Trusted OS is installed after this step, in phase 3 or 5 though it may be upgraded later. Trusted Applications may be installed together with or after the Trusted OS, either in step 3 or in step 5 – for this latter case, they may have been linked with the Trusted OS in step 4. If the TOE supports TEE Debug functionalities, the flag to indicate whether the functionality is enabled on the TEE and the Debug credentials such as a Debug authentication key are set in phase 3 or 5.

The TOE delivery point establishes the limits of the evaluation: The delivery point can range from phase 3 to phase 5, but must necessarily follow the setting of the root of trust of the TEE storage services, of the TEE unique identifier, of the Debug enabled flag and the Debug credentials (if TEE Debug PP-module is included), and the Trusted OS installation:

- The security of the environments, processes and procedures before the delivery point is evaluated according to the EAL 2 through the ALC assurance class
- The security of the environments processes and procedures from the delivery point up to end of phase 5 are covered through the AGD assurance class by organizational security policies and security objectives for the environment
- The security of the end-usage environment is covered by the TOE security functionalities and by security objectives for the environment.

Figure 2-4, together with Table 4, represents one possible instantiation of the six phases. The table presents the actors involved in the different life cycle phases. Note that actors may delegate operations to other entities provided the overall security level is met.

For the sake of readability, the diagram and table are not meant to cover all possibilities. Other flows, consisting of the six same phases, but with different ownership of the steps represented, are possible. Cases where the TEE runs on a discrete separate processor, or where the TEE is installed by the chipset manufacturer in case the device manufacturer merely integrates a turn-key platform that the chipset manufacturer provides, or on the contrary where the device manufacturer is fully responsible for TEE integration, can result in different flows.

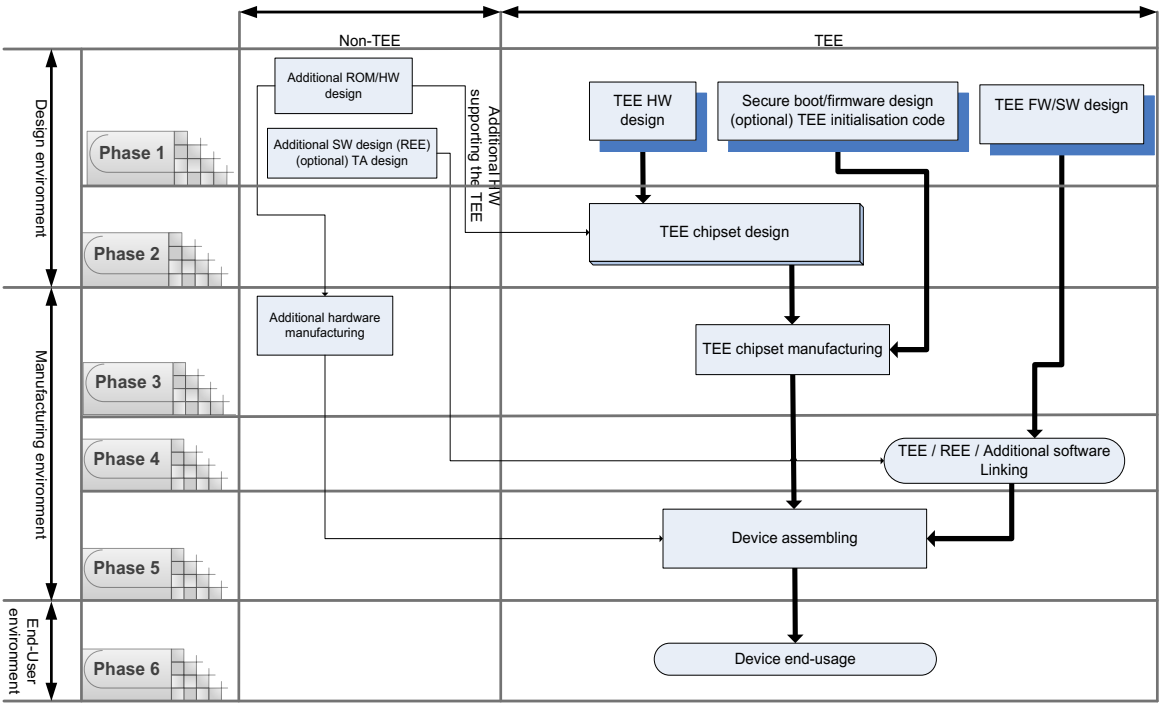

Figure 2-4 –Life Cycle of TEE-enabled Device

Phases	Actors
1 & 2: Firmware / Software / Hardware design	The TEE software developer • Is in charge of TEE software development and testing compliant with GlobalPlatform specifications • May also develop the TEE initialization code that instantiates/initializes the TEE (e.g. part of the secure boot code) • 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. He may also design Trusted Applications that he 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.

Phases	Actors
	The silicon vendor designs the ROM code and the secure portion of the TEE chipset. If the silicon vendor is not designing the full TEE hardware, the silicon vendor integrates (and potentially augments) the TEE hardware designed by the TEE hardware designer(s).
3: TEE manufacturing	The silicon vendor produces the TEE chipset and enables, sets or seeds the root of trust of the TEE.
4: Software manufacturing	The device manufacturer is responsible for the integration, validation and preparation of the software to load in the product that will include the TEE, any pre-installed Trusted Application, and additional software required to use the product (e.g. REE, Client Applications).
5: 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.
6: End-usage phase	The end user gets a device ready for use. The Trusted Applications manager is responsible for the loading, installation, and removal of Trusted Applications post-issuance.

Table 4: Actors in the Device Life Cycle

Application Note: Security Targets shall describe the actual TOE life cycle, identify the actors and development/manufacturing sites involved; they shall identify the actual integration points of the components (Trusted OS, root of trust, TAs) into the device, as well as the actual delivery point of the TOE, and precise the process for setting the root of trust of the TEE storage services and the phase in which it occurs.

Security targets shall also identify the TOE and the components that are delivered with the TOE if any, e.g. the standard OS, pre-installed Trusted Applications or Client Applications. If the TOE provides TA management functionality (i.e. installation of TAs in phase 6 or in general after the delivery point), which is not in the scope of this Protection Profile, it must be described in the ST as well.

3 Conformance Claims and Consistency Rationale

This document uses modular PP methodology [PP-MOD] and includes a base-PP and two PP-modules. This section applies to the base-PP and to the TEE Time and Rollback and TEE Debug PP-modules.

3.1 Conformance Claim to CC

This base Protection Profile is CC Part 2 [CC2] extended and CC Part 3 [CC3] extended. The CC Part 2 is extended with the security functional components FCS_RNG.1 Random numbers generation and FPT_INI.1 TSF initialisation.

The CC Part 3 is extended with the security assurance component AVA_TEE.2 Low TEE vulnerability analysis. Annex A.2 explains the relationship between AVA_TEE.2 and AVA_VAN.2. Both SARs relate to vulnerability analysis. They only differ in the ratings defined in the attack potential quotation grids used for each AVA components. As both AVA components are claimed, products evaluated in conformance with this PP will have to be assessed according to the two quotation grids.

The TEE Time and Rollback and TEE Debug PP-modules are CC Part 2 [CC2] conformant.

3.2 Conformance Claim to a Package

The minimum assurance level for the evaluation of a TOE conformant to this PP is EAL 2 augmented with AVA_TEE.2, defined in section 6.1.4.

This conformance claim also applies to the PP-configurations defined in this document.

3.3 Conformance Claim of the PP

This PP does not claim conformance to any another PP.

3.4 Conformance Claim to the PP

The conformance to this PP, required for the Security Targets and Protection Profiles claiming conformance to it, is strict as defined in CC Part 1 [CC1].

This conformance claim also applies to the PP-configurations defined in this document.

3.5 Consistency Rationale for the PP-Modules

3.5.1 TEE Time and Rollback PP-Module

The TEE Time and Rollback PP-Module is intended to be used only with the base-PP of this document. It complements the TEE functionalities defined in section 2.3.1 that form the base-PP. It defines a new functionality which is the monotonic TA persistent time and extends the consistency verification on trusted storage, TA code and configuration data to integrity verification.

The PP-module does not add any assumption nor OSP nor security objective of environment.

The PP-module adds two threats and corresponding security objectives related to persistent time and rollback. The PP-module adds four new SFRs, including one FDP_SDI.2/Rollback that extends FDP_SDI.2 of the base-PP to the integrity property instead of only consistency property.

The unions of the SPD, the objectives and the security functional requirements from the base PPs and from the PP-module do not lead to a contradiction.

3.5.2 TEE Debug PP-Module

The TEE Debug PP-Module is intended to be used only with the base-PP of this document. It complements the TEE functionalities defined in section 2.3.1 that form the base-PP. It defines the possibility for the TEE Debug Administrator to be granted access to the Debug features, after authentication.

The PP-module does not add any assumption nor OSP nor security objective of environment.

The PP-module adds one new threat and a corresponding security objective related to access control to the debug interface. The PP-module adds six new SFRs for access control.

The unions of the SPD, the objectives and the security functional requirements from the base PPs and from the PP-module do not lead to a contradiction.

Furthermore, debug features are independent from the functionalities defined in the TEE Time and Rollback PP-Module. Both PP-modules can be used independently.

4 Security Problem Definition

This chapter introduces the security problem addressed by the TEE and its operational environment. The operational environment stands for the TEE integration and maintenance environment and the TA development environment. The security problem consists of the threats the TEE-enabled devices may face in the field, the assumptions on its operational environment and the organizational policies that have to be implemented by the TEE or within the operational environment.

4.1 Assets

This section presents the assets of the TOE and their properties: authenticity, consistency, integrity, confidentiality, monotonicity, randomness, atomicity, read-only and device binding (cf. Section 1.4 for definitions).

4.1.1 TEE base-PP

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

Random Number Generator.

Properties: unpredictable random numbers, sufficient entropy.

TA code

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

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

TA data and keys

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

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

TA instance time

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

Properties: monotonicity.

TEE runtime data

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

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

TEE persistent data

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

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

TEE firmware

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

Properties: authenticity, integrity.

TEE initialization code and data

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

Properties: integrity.

TEE storage root of trust

The root of trust of the TEE storage that is used to bind the stored data and keys to the TEE. This data is typically stored in the Trusted OTP memory of the TEE.

Properties: integrity and confidentiality.

Application Note:

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

4.1.2 TEE Time and Rollback PP-Module

The assets of this PP-module extend the assets of the TEE base PP as follows:

- "TA persistent time" is a new asset
- "TA data and keys_module", "TA code_module" and "TEE data_module" are the same assets as in the base PP but with both consistency and integrity properties. This means that the TOE has to provide full rollback protection.

TA persistent time

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

Properties: monotonicity.

TA data and keys_module

Data and keys managed and stored by TA using 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.

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

Application Note:

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

TA code_module

The code of the installed Trusted Applications.

Properties: authenticity, consistency and integrity.

Application Note:

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

TEE data_module

Persistent TEE data, including TEE keys.

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

TEE rollback detection data

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

Properties: integrity.

4.1.3 TEE Debug PP-Module

The asset of this PP-module extends the assets of the TEE base PP by providing a new cryptographic key.

TEE debug authentication key

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

Properties: integrity and confidentiality

4.2 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.

4.2.1 TEE base-PP

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.

4.2.2 TEE Debug PP-Module

TEE Debug Administrator

The TEE Debug Administrator or actor acting on his behalf that can be granted access to TEE debug features.

4.3 Threats

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

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

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

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

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

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

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.

4.3.1 TEE base-PP

The following threats apply to any TEE.

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 cleartext, thus disclosing sensitive TA and TEE data and potentially allowing the attacker to mount other attacks.

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

Application Note:

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

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

T.IMPERSONATION

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

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

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

T.ROGUE_CODE_EXECUTION

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

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

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

Application Note:

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

T.PERTURBATION

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

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

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

Application Note:

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

T.RAM

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

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

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

Application Note:

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

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

T.RNG

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

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

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

T.SPY

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

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

Application Note:

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

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

T.TEE_FIRMWARE_DOWNGRADE

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

Assets threatened directly (integrity): TEE firmware.

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

T.STORAGE_CORRUPTION

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

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

Application Note:

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

4.3.2 TEE Time and Rollback PP-Module

The following two threats apply to TEEs implementing trusted storage and TA persistent time integrity (also called anti-rollback property).

Moreover, the standard threat T.STORAGE_CORRUPTION is no longer linked to OE.ROLLBACK but to O.ROLLBACK_PROTECTION.

T.ROLLBACK

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

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

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

Application Note:

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

T.TA_PERSISTENT_TIME_ROLLBACK

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

Assets threatened directly (integrity): TA persistent time.

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

Application Note:

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

4.3.3 TEE Debug PP-Module

T.ABUSE_DEBUG

An attacker manages to be granted access to TEE Debug features, 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:

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

4.4 Organizational Security Policies

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

4.4.1 TEE base-PP

The following policies apply to any TEE.

OSP.INTEGRATION_CONFIGURATION

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

Application Note:

The security target shall reference the applicable TEE guidelines, in particular the operational guidance that fulfills AGD_OPE.1 requirements.

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).

4.4.2 TEE Time and Rollback PP-Module

There is no additional policy in the Time and Rollback PP-Module.

4.4.3 TEE Debug PP-Module

There is no additional policy in the TEE Debug PP-Module.

4.5 Assumptions

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

4.5.1 TEE base-PP

The following assumptions hold on the TEE operational environment.

A.PROTECTION_AFTER_DELIVERY

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

Application Note:

The certificate is valid only when the guidelines 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.

The security target shall reference the applicable TEE guidelines, in particular the operational guidance that fulfills AGD_OPE.1 requirements.

A.ROLLBACK

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

A.TA_DEVELOPMENT

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

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

4.5.2 TEE Time and Rollback PP-Module

There is no additional assumption in the Time and Rollback PP-Module. Moreover, the assumption A.TA_ROLLBACK is discarded since the TOE enforces anti-rollback protection.

4.5.3 TEE Debug PP-Module

There is no additional assumption in the TEE Debug PP-Module.

5 Security Objectives

5.1 Security Objectives for the TOE

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

5.1.1 TEE base-PP

The following security objectives apply to any TEE.

O.CA_TA_IDENTIFICATION

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

Application Note:

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

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

O.KEYS_USAGE

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

O.TEE_ID

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

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

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

5.1.2 TEE Time and Rollback PP-Module

The following two objectives apply to TEEs implementing trusted storage and TA persistent time integrity (also called anti-rollback property).

O.ROLLBACK_PROTECTION

The TEE shall prevent unauthorized rollback by:

- monitoring integrity violation of TEE persistent data, TA data or keys, or TA code;
- react accordingly so that the security is always enforced.

Application Note:

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

O.TA_PERSISTENT_TIME

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

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

5.1.3 TEE Debug PP-Module

O.DEBUG

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

5.2 Security Objectives for the Operational Environment

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

5.2.1 TEE base-PP

The following security objectives apply to any TEE 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 TEE provider that fulfill the requirements set in GlobalPlatform TEE specifications and state all the security requirements for the device manufacturer issued from the TOE evaluation.

OE.PROTECTION_AFTER_DELIVERY

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

Application Note:

The certificate is valid only when the guides are applied. For instance, for installation, 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.

5.2.2 TEE Time and Rollback PP-Module

Anti-rollback protection is enforced by the TOE (cf. O.ROLLBACK_PROTECTION). Henceforth the objective for the operational environment OE.ROLLBACK from the base Protection Protection is discarded when this PP-Module is used.

5.2.3 TEE Debug PP-Module

There is no additional security objective for the Operational Environment in the TEE Debug PP-Module.

5.3 Security Objectives Rationale

5.3.1 Threats

5.3.1.1 TEE base-PP

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

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

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

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

- O.INITIALIZATION ensures that the TEE security functionality is correctly initialized and the integrity of TEE firmware
- O.OPERATION ensures correct operation of the security functionality
- O.RUNTIME_CONFIDENTIALITY covers runtime TEE data which might influence the behavior of the TEE
- O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified
- O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime
- O.TEE_DATA_PROTECTION covers persistent TEE data which might influence the behavior of the TEE
- O.TRUSTED_STORAGE ensures protection of the storage from which code might be imported
- OE.INTEGRATION_CONFIGURATION covers the import of foreign code in a phase other than the end-user phase
- OE.PROTECTION_AFTER_DELIVERY covers the import of foreign code in a phase other than the end-user phase.

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

- O.INITIALIZATION 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).

Additional rationale specific to the Time and Rollback PP-Module:

- O.TA_PERSISTENT_TIME ensures the reliability of persistent time stamps

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

- O.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.

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.

Rationale specific to the Time and Rollback PP-Module:

- The threat T.STORAGE_CORRUPTION is not linked to OE.ROLLBACK but to O.ROLLBACK_PROTECTION.

5.3.1.2 TEE Time and Rollback PP-Module

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

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

5.3.1.3 TEE Debug PP-Module

T.ABUSE_DEBUG The following objective ensures protection against abuse of debug functionality:

- O.DEBUG ensure authentication of TEE Debug Administrator before granting access to TEE debug features.

5.3.2 Organizational Security Policies

5.3.2.1 TEE base-PP

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

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

5.3.3 Assumptions

5.3.3.1 TEE base-PP

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

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

Rationale specific to the Time and Rollback PP-Module: the assumption does not apply to TEEs implementing the Time and Rollback PP-Module.

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

5.3.4 SPD and Security Objectives

Threats	Security Objectives	Rationale
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	Section 2.3.1

Threats	Security Objectives	Rationale
T.CLONE	O.TEE_ID, O.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, O.RNG	Section 2.3.1
T.FLASH_DUMP	O.TRUSTED_STORAGE	Section 2.3.1
T.IMPERSONATION	O.CA_TA_IDENTIFICATION, O.OPERATION, O.RUNTIME_INTEGRITY	Section 2.3.1
T.ROGUE_CODE_EXECUTION	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY, O.TA_AUTHENTICITY, O.INITIALIZATION	Section 2.3.1
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	Section 2.3.1
T.RAM	O.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION	Section 2.3.1
T.RNG	O.INITIALIZATION, O.RNG, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY	Section 2.3.1
T.SPY	O.RUNTIME_CONFIDENTIALITY, O.TA_ISOLATION, O.TEE_ISOLATION, O.TRUSTED_STORAGE	Section 2.3.1
T.TEE_FIRMWARE_DOWNGRADE	O.INITIALIZATION, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY	Section 2.3.1

Threats	Security Objectives	Rationale
T.STORAGE_CORRUPTION	O.OPERATION, O.ROLLBACK_PROTECTION, OE.ROLLBACK, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, O.TA_AUTHENTICITY, O.INITIALIZATION	Section 2.3.1
T.ROLLBACK	O.ROLLBACK_PROTECTION	Section 2.3.1
T.TA_PERSISTENT_TIME_ROLLBACK	O.TA_PERSISTENT_TIME	Section 2.3.1
T.ABUSE_DEBUG	O.DEBUG	Section 2.3.1

Table 5 Threats and Security Objectives - Coverage

Security Objectives	Threats
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
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
O.ROLLBACK_PROTECTION	T.STORAGE_CORRUPTION, T.ROLLBACK
O.TA_PERSISTENT_TIME	T.PERTURBATION, T.TA_PERSISTENT_TIME_ROLLBACK
O.DEBUG	T.ABUSE_DEBUG

Security Objectives	Threats
OE.INTEGRATION_CONFIGURATION	T.ROGUE_CODE_EXECUTION, T.TEE_FIRMWARE_DOWNGRADE
OE.PROTECTION_AFTER_DELIVERY	T.ROGUE_CODE_EXECUTION, T.TEE_FIRMWARE_DOWNGRADE
OE.ROLLBACK	T.STORAGE_CORRUPTION
OE.SECRETS	
OE.TA_DEVELOPMENT	T.ABUSE_FUNCT

Table 6 Security Objectives and Threats - Coverage

Organisational Security Policies	Security Objectives	Rationale
OSP.INTEGRATION_CONFIGURATION	OE.INTEGRATION_CONFIGURATION	Section 2.3.2
OSP.SECRETS	OE.SECRETS	Section 2.3.2

Table 7 OSPs and Security Objectives - Coverage

Security Objectives	Organisational Security Policies
O.CA_TA_IDENTIFICATION	
O.KEYS_USAGE	
O.TEE_ID	
O.INITIALIZATION	
O.INSTANCE_TIME	
O.OPERATION	
O.RNG	
O.RUNTIME_CONFIDENTIALITY	
O.RUNTIME_INTEGRITY	
O.TA_AUTHENTICITY	
O.TA_ISOLATION	
O.TEE_DATA_PROTECTION	
O.TEE_ISOLATION	
O.TRUSTED_STORAGE	
O.ROLLBACK_PROTECTION	
O.TA_PERSISTENT_TIME	
O.DEBUG	
OE.INTEGRATION_CONFIGURATION	OSP.INTEGRATION_CONFIGURATION

Security Objectives	Organisational Security Policies
OE.PROTECTION_AFTER_DELIVERY	
OE.ROLLBACK	
OE.SECRETS	OSP.SECRETS
OE.TA_DEVELOPMENT	

Table 8 Security Objectives and OSPs - Coverage

Assumptions	Security Objectives for the Operational Environment	Rationale
A.PROTECTION_AFTER_DELIVERY	OE.PROTECTION_AFTER_DELIVERY	Section 2.3.3
A.ROLLBACK	OE.ROLLBACK	Section 2.3.3
A.TA_DEVELOPMENT	OE.TA_DEVELOPMENT	Section 2.3.3

Table 9 Assumptions and Security Objectives for the Operational Environment - Coverage

Security Objectives for the Operational Environment	Assumptions
OE.INTEGRATION_CONFIGURATION	
OE.PROTECTION_AFTER_DELIVERY	A.PROTECTION_AFTER_DELIVERY
OE.ROLLBACK	A.ROLLBACK
OE.SECRETS	
OE.TA_DEVELOPMENT	A.TA_DEVELOPMENT

Table 10 Security Objectives for the Operational Environment and Assumptions - Coverage

6 Extended Requirements

6.1 Extended Families

6.1.1 Extended Family FCS_RNG - Generation of random numbers

6.1.1.1 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.

6.1.1.2 Extended Components

6.1.1.3 Extended Component FCS_RNG.1

6.1.1.4 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.

6.1.1.5 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.

6.1.2 Extended Family FPT_INI - TSF initialisation

6.1.2.1 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.

6.1.2.2 Extended Components

6.1.2.3 Extended Component FPT_INI.1

6.1.2.4 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.

6.1.2.5 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.

6.1.3 Extended Family AVA_TEE - Vulnerability analysis of TEE

6.1.3.1 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.

6.1.3.2 Extended Components

6.1.3.3 Extended Component AVA_TEE.2

6.1.3.4 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.

6.1.3.5 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

7 Security Requirements

This chapter introduces the security problem addressed by the TEE and its operational environment. The operational environment stands for the TEE integration and maintenance environment and the TA development environment. The security problem consists of the threats the TEE-enabled devices may face in the field, the assumptions on its operational environment and the organizational policies that have to be implemented by the TEE or within the operational environment.

7.1 Security Functional Requirements

This chapter provides the set of Security Functional Requirements (SFRs) the TOE has to enforce in order to fulfill the security objectives.

7.1.1 TEE base-PP

This Protection Profile uses the following security functional components defined in CC Part 2 [CC2]:

- FAU_ARP.1 Security alarms
- FAU_SAR.1 Audit review
- FAU_STG.1 Audit event storage
- FCS_COP.1 Cryptographic operation
- FIA_ATD.1 User attribute definition
- FIA_UID.2 User identification before any action
- FIA_USB.1 User-subject binding
- FDP_ACC.1 Subset access control
- FDP_ACF.1 Security attribute based access control
- FDP_IFC.2 Complete information flow control
- FDP_IFF.1 Simple security attributes
- FDP_ITT.1 Basic internal transfer protection
- FDP_RIP.1 Subset residual information protection
- FDP_ROL.1 Basic rollback
- FDP_SDI.2 Stored data integrity monitoring and action
- FMT_MSA.1 Management of security attributes
- FMT_MSA.3 Static attribute initialization
- FMT_SMR.1 Security roles
- FMT_SMF.1 Management functions
- FPT_FLS.1 Failure with preservation of secure state
- FPT_ITT.1 Basic internal TSF data transfer protection
- FPT_STM.1 Reliable time stamps
- FPT_TEE.1 Testing of external entities

Moreover, the following extended security functional components, defined in Chapter 6, are used:

- FCS_RNG.1 Random numbers generation

- FPT_INI.1 TSF initialisation

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

Objects stand for passive entities inside the TOE:

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

Cryptographic objects are a special kind of TEE objects:

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

Information stands for data exchanged between subjects:

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

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

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

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

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

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

Other operations:

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

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

Runtime Data Information Flow Control SFP:

- Purpose: To control the flow of runtime data from and to executable entities and memory. This policy contributes to 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.

Application note: The Security Target writer shall fill in the SFR open assignments and perform the selections that are appropriate for their product. The TOE Summary Specification (TSS) shall describe how the product implements the instantiated requirements. Note that the requirements may imply supporting security functionality, for instance:

- Authentication/signature and encryption/decryption of storage spaces located in external memory (cf. FDP_ACF.1/Trusted Storage)
- Binding of Client Applications with TA sessions (cf. FIA_USB.1)
- Verification of the client_identity when the client is a TA (cf. FIA_USB.1)
- Binding of trusted storage with the Storage Root of Trust OB.SRT (cf. FDP_ACF.1/Trusted Storage)
- Configuration of TEE resources shared with the REE (cf. FDP_IFF.1/Runtime)
- Secure state definition and entering process upon failure management (cf. FPT_FLS.1).

7.1.1.1 Identification

FIA_ATD.1 User attribute definition

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

Application Note:

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

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

FIA_UID.2 User identification before any action

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

Application Note:

User stands for Client Application or Trusted Application.

FIA_USB.1 User-subject binding

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

- **Client (CA or TA) identity is codified into the client_identity of the requested TA session**
- **[assignment: list of user security attributes]**.

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

- **If the client is a TA, then the client_identity must be equal to the TA_identity of the TA subject that is the client**
- **[assignment: rules for the initial association of attributes]**.

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

- **No modification of client_identity is allowed after initialization**
- **[assignment: rules for the changing of attributes]**.

Application Note:

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

FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles

- **TSF**
- **TA_User**
- **[assignment: the authorized identified roles]**.

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.

7.1.1.2 Confidentiality, Integrity and Isolation

FDP_IFC.2/Runtime Complete information flow control

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

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

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

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

Application Note:

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

FDP_IFF.1/Runtime Simple security attributes

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

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

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

FDP_IFF.1.3/Runtime The TSF shall enforce the **[assignment: additional information flow control SFP rules]**.

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

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

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

Application Note:

- The access rights configuration managed by S.RAM_UNIT shall ensure that RAM addressable units used to TSF data are appropriately protected (in integrity for TEE firmware, in integrity and confidentiality for TEE runtime data)
- RAM units can span over several volatile memories, for example, on-chip RAM, off-chip RAM, registers
- TEE-dedicated RAM units may hold copies of the content of temporary memory references passed by the REE

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".

7.1.1.3 Cryptography

FCS_COP.1 Cryptographic operation

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

Application Note:

The Security Target shall provide in this SFR cryptographic operations used within the TOE for:

- verifying the authenticity of TEE firmware and TA code
- protecting the consistency and confidentiality of Trusted Storage data. These operations are based on the TEE storage root of trust key. The ST writer may choose to include in other iteration of FCS_COP.1 additional cryptographic operations, for instance operations provided by Internal API to the TA.

FDP_ACC.1/TA_keys Subset access control

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

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

FDP_ACF.1/TA_keys Security attribute based access control

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

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

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

FDP_ACF.1.3/TA_keys The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly authorise access of subjects to objects]**.

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)**
- **[assignment: rules, based on security attributes, that explicitly deny access of subjects to objects]**.

Application Note:

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

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

FMT_MSA.1/TA_keys Management of security attributes

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

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

FMT_MSA.3/TA_keys Static attribute initialisation

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

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

7.1.1.4 Initialization, Operation and Firmware Integrity

FAU_ARP.1 Security alarms

FAU_ARP.1.1 The TSF shall take **[assignment: list of actions]** 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: **[assignment: associated actions]**.
- detection of TEE firmware integrity violation: **[assignment: associated actions]**.
- **[assignment: list of implementation-dependent potential security violations and associated actions]**.

FDP_SDI.2 Stored data integrity monitoring and action

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

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

Refinement:

- Upon detection of authenticity or consistency errors in TEE runtime data or TEE persistent data, the TSF shall **[assignment: 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**

- **[assignment: action that does not depend on the compromised data]**
- **[assignment: other actions to be taken]**.

Application Note:

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

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

FPT_FLS.1 Failure with preservation of secure state

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

- **Device binding failure**
- **Cryptographic operation failure**
- **Invalid CA requests, in particular bad-formed requests**
- **Panic states (as defined in [IAPI], Section 2.2.3)**
- **TA code, TA data or TA keys authenticity or consistency failure**
- **TEE data (in particular TA properties, TEE keys and all security attributes) authenticity or consistency failure**
- **TEE firmware integrity failure**
- **TEE initialization failure**
- **Unexpected commands in the current TEE state**
- **[assignment: list of types of failures in the TSF]**.

Application Note:

- Device binding failure occurs when (part of) the stored data has not been bound by the same TEE
- The ST writer shall define the characteristics of the secure state. In particular, the transition between a failure state and the secure state shall protect TEE and user data and keys confidentiality.

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.

Application Note:

Firmware downgrade verification has to rely on data residing on the TOE, for instance on One Time Programmable (OTP) memories or EEPROM.

FMT_SMF.1 Specification of Management Functions

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

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

FPT_TEE.1 Testing of external entities

FPT_TEE.1.1 The TSF shall run a suite of tests **prior execution and [assignment: other conditions]** 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**.

7.1.1.5 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 unauthorised deletion.

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

Application Note:

The audit record in this SFR refer to the TEE identifier. This unique identifier is stored on the TOE before TOE delivery. It can be generated on-TEE or off-TEE (the Security Target shall precise the generation method).

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

The Security Target shall indicate in which type of persistent memory it is stored.

7.1.1.6 Instance Time

FPT_STM.1/Instance time Reliable time stamps

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

Refinement:

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

Application Note:

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

7.1.1.7 Random Number Generator

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]**.

Application Note:

The ST writer shall perform the missing operation in the elements FCS_RNG.1.1 and FCS_RNG_1.2. The ST writer should define the quality of the generated random numbers using for instance the Min-entropy or Shannon entropy. The assignment of a quality metric shall ensure sufficient randomness of the random numbers near to the uniform distributed random variables. The evaluation of the random number generator shall follow a recognized methodology, e. g. AIS31. This SFR is also used to ensure statistical uniqueness of the TEE identification if it is generated on the TEE.

7.1.1.8 Trusted Storage

FDP_ACC.1/Trusted Storage Subset access control

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

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

FDP_ACF.1/Trusted Storage Security attribute based access control

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

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

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

FDP_ACF.1.3/Trusted Storage The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly authorise access of subjects to objects]**.

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**
- **[assignment: rules, based on security attributes, that explicitly deny access of subjects to objects]**.

FDP_ROL.1/Trusted Storage Basic rollback

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

FDP_ROL.1.2/Trusted Storage The TSF shall permit operations to be rolled back within the **[assignment: boundary limit to which rollback may be performed]**.

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 initialisation

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

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

FDP_ITT.1/Trusted Storage Basic internal transfer protection

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

7.1.2 TEE Time and Rollback PP-Module

The following security functional components defined in CC Part 2 [CC2] are used in this PP-module:

- FDP_SDI.2 Stored data integrity monitoring and action
- FPT_FLS.1 Failure with preservation of secure state
- FPT_STM.1 Reliable time stamps
- FMT_MTD.1 Management of TSF data
- FMT_SMF.1 Specification of Management Functions.

7.1.2.1 Rollback Protection

FDP_SDI.2/Rollback Stored data integrity monitoring and action

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

Refinement:

The TSF shall monitor TEE rollback detection data, TEE runtime data, TEE persistent data, TA data and keys and TA code stored in containers controlled by the TSF for **integrity errors** on all objects, based on the following attributes: **[assignment: attributes of TEE rollback detection data, TEE runtime data, TEE persistent data, TA data and keys and TA code]**.

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

Refinement:

- Upon detection of integrity errors in TEE rollback detection data, TEE runtime data or TEE persistent data, the TSF shall **behave in a manner that does not depend on the compromised data**
- Upon detection of TA code integrity errors, the TSF shall **abort the execution of the TA instance**
- Upon detection of TA data or TA keys integrity errors, the TSF shall
  - **Not provide any compromised data,**
  - **Behave in a manner that does not depend on the compromised data**
- **[assignment: other actions to be taken]**.

Application Note:

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

FPT_FLS.1/Rollback Failure with preservation of secure state

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

- **TA code and data integrity failure**
- **TEE persistent data integrity failure**.

Application Note:

This requirement is a complement to FPT_FLS.1.

7.1.2.2 TA Persistent Time

FPT_STM.1/Persistent Time Reliable time stamps

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

Refinement:

The TSF shall be able to provide time stamps to TA instances such that:

- Time stamps are persistent over TEE reset
- Time stamps are monotonic between two 'time setting' operations performed by any instance of the TA.

The TSF shall invalidate any persistent time that does not meet the monotonicity property.

Application Note:

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

FMT_MTD.1/Persistent Time Management of TSF data

FMT_MTD.1.1/Persistent Time The TSF shall restrict the ability to **perform a 'time setting' operation on** the **TA persistent time** to **any instance of the TA**.

Application Note:

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

FMT_SMF.1/Persistent Time Specification of Management Functions

FMT_SMF.1.1/Persistent Time The TSF shall be capable of performing the following management functions: **'time setting' operation for TA persistent time**.

Application Note:

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

7.1.3 TEE Debug PP-Module

The following security functional components defined in CC Part 2 [CC2] are used in this PP-module:

- FIA_UID.2 User identification before any action
- FIA_UAU.2 User authentication before any action
- FIA_UAU.6 Re-authenticating
- FIA_ATD.1 User attribute definition
- FIA_USB.1 User-subject binding
- FCS_COP.1 Cryptographic operation
- FDP_ACC.1 Subset access control
- FDP_ACF.1 Security attribute based access control
- FMT_SMR.1 Security roles.

All these SFRs defined in this section only concern the Debug functionality

The additional User introduced by this PP-module is:

- TEE Debug Administrator

The additional subject introduced by this PP-module is:

- S.DEBUG: the debug interface, with security attributes "enabled" (True/False) to state whether this feature is available on the TEE (attribute set before TOE delivery and not modifiable afterwards) and "authenticated" (True/False) to state whether the TEE Debug Administrator has been authenticated.

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

- OP.AUTHENTICATE: activation of the debug feature by TEE Debug Administrator authentication
- OP.DEBUG: debug operations.

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

Debug access control SFP:

- Purpose: To control the access to debug facilities of the TEE.
- Subjects: S.DEBUG
- Objects: all
- Security attributes: S.DEBUG.enabled, S.DEBUG.authenticated
- Operations: OP.AUTHENTICATE, OP.DEBUG

- SFR intances: FDP_ACC.1/Debug, FDP_ACF.1/Debug.

FDP_ACC.1/Debug Subset access control

FDP_ACC.1.1/Debug The TSF shall enforce the **Debug access control SFP** on

- **Subjects: S.DEBUG**
- **Objects: all objects**
- **Operations: OP.ACTIVATE, OP.DEBUG**.

FDP_ACF.1/Debug Security attribute based access control

FDP_ACF.1.1/Debug The TSF shall enforce the **Debug access control SFP** to objects based on the following:

- **S.DEBUG.enabled, S.DEBUG.authenticated**
- **[assignment: list of subjects and objects controlled under the indicated SFP, and for each, the SFP-relevant security attributes, or named groups of SFP-relevant security attributes]**.

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

- **OP.AUTHENTICATE is allowed if the following conditions hold:**
  - **The operation is performed by S.DEBUG**
  - **The debug interface is enabled (S.DEBUG.enabled = True)**
- **OP.DEBUG on all objects is allowed if the following conditions hold:**
  - **The operation is performed by S.DEBUG**
  - **The debug interface is enabled (S.DEBUG.enabled = True)**
  - **The TEE Debug Administrator is authenticated (S.DEBUG.authenticated = True)**
- **[assignment: rules governing access among controlled subjects and controlled objects using controlled operations on controlled objects]**.

FDP_ACF.1.3/Debug The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly authorise access of subjects to objects]**.

FDP_ACF.1.4/Debug The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[assignment: rules, based on security attributes, that explicitly deny access of subjects to objects]**.

FCS_COP.1/Debug Cryptographic operation

FCS_COP.1.1/Debug The TSF shall perform **authentication of the TEE Debug Administrator or the actor acting on his behalf** in accordance with a specified cryptographic algorithm

[assignment: cryptographic algorithm] and cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignment: list of standards]**.

FMT_SMR.1/Debug Security roles

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

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

Application Note:

The TEE Debug Administrator is not intended to be the end-user, but someone involved in the life-cycle of the product and who has access to the debug credential set during phase 3 or 5.

FIA_UID.2/Debug User identification before any action

FIA_UID.2.1/Debug [Editorially Refined] The TSF shall require each user to be successfully identified before allowing any other TSF-mediated **debug** actions on behalf of that user.

FIA_ATD.1/Debug User attribute definition

FIA_ATD.1.1/Debug The TSF shall maintain the following list of security attributes belonging to individual users: **S.DEBUG.enabled, S.DEBUG.authenticated**.

FIA_USB.1/Debug User-subject binding

FIA_USB.1.1/Debug The TSF shall associate the following user security attributes with subjects acting on the behalf of that user: **S.DEBUG.enabled, S.DEBUG.authenticated**.

FIA_USB.1.2/Debug The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **S.DEBUG.authenticated is False**.

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

- **S.DEBUG.authenticated is set to True after TEE Debug Administrator successful authentication**
- **S.DEBUG.authenticated is set to False when the authentication is lost, for instance after power-off (cf. rules of FIA_UAU.6)**
- **[assignment: rules for the changing of attributes]**.

FIA_UAU.2/Debug User authentication before any action

FIA_UAU.2.1/Debug [Editorially Refined] The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated **debug** actions on behalf of that user.

FIA_UAU.6/Debug Re-authenticating

FIA_UAU.6.1/Debug The TSF shall re-authenticate the user under the conditions

- **after TEE power-off**
- **[assignment: list of conditions under which re-authentication is required]**.

7.2 Security Assurance Requirements

The Protection Profile provides a set of Security Assurance Requirements (SARs) for the PP which 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, defined in Annex A.1.

As both AVA_VAN.2 and AVA_TEE.2 are selected in the augmented EAL, the evaluator will have to perform two attack quotations according to the quotation grids associated with each of these SARs.

7.3 Security Requirements Rationale

7.3.1 Objectives

7.3.1.1 Security Objectives for the TOE

7.3.1.2 TEE base-PP

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.

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.

Rationale specific to the Time and Rollback PP-Module:

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

Rationale specific to the Debug PP-Module:

- FDP_ACC.1/Debug, FDP_ACF.1/Debug, FMT_SMR.1/Debug, FIA_UID.2/Debug, FIA_UAU.2/Debug, FIA_UAU.6/Debug, FIA_ATD.1/Debug and FIA_USB.1/Debug state the debug access policy, which grants access to the debug facilities of the TEE if this feature is not disabled

O.RNG The requirement FCS_RNG.1 directly 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.

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.

7.3.1.3 TEE Time and Rollback PP-Module

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

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

O.TA_PERSISTENT_TIME The following requirements fulfill the objective:

- FPT_STM.1/Persistent Time states the persistent time reliability conditions expected from the TEE
- FMT_MTD.1/Persistent Time states the roles that can perform 'time-setting' operations
- FMT_SMF.1/Persistent Time states the existence of a 'time-setting' managemenent function.

7.3.1.4 TEE Debug PP-Module

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

- FDP_ACC.1/Debug, FDP_ACF.1/Debug, FMT_SMR.1/Debug, FIA_UID.2/Debug, FIA_UAU.2/Debug, FIA_UAU.6/Debug, FIA_ATD.1/Debug and FIA_USB.1/Debug state the debug access policy, which grants access to the TEE Debug Administrator only
- FCS_COP.1/Debug allows to specify the cryptographic operations used for authenticating TEE Debug Administrator.

7.3.2 Rationale tables of Security Objectives and SFRs

Security Objectives	Security Functional Requirements	Rationale
O.CA_TA_IDENTIFICATION	FIA_ATD.1, FIA_UID.2, FIA_USB.1	Section 4.3.1
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	Section 4.3.1
O.TEE_ID	FAU_SAR.1, FCS_RNG.1, FPT_INI.1, FAU_STG.1	Section 4.3.1
O.INITIALIZATION	FPT_FLS.1, FPT_INI.1, FCS_COP.1	Section 4.3.1
O.INSTANCE_TIME	FPT_STM.1/Instance time	Section 4.3.1

Security Objectives	Security Functional Requirements	Rationale
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_ACC.1/Debug, FDP_ACF.1/Debug, 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, FMT_SMR.1/Debug, FIA_UID.2/Debug, FIA_ATD.1/Debug, FIA_USB.1/Debug, FIA_UAU.2/Debug, FIA_UAU.6/Debug	Section 4.3.1
O.RNG	FCS_RNG.1	Section 4.3.1
O.RUNTIME_CONFIDENTIALITY	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_ITT.1/Runtime, FDP_RIP.1/Runtime, FPT_ITT.1/Runtime	Section 4.3.1
O.RUNTIME_INTEGRITY	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_ITT.1/Runtime, FPT_ITT.1/Runtime, FDP_SDI.2	Section 4.3.1
O.TA_AUTHENTICITY	FDP_SDI.2, FCS_COP.1, FPT_TEE.1	Section 4.3.1
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	Section 4.3.1
O.TEE_DATA_PROTECTION	FDP_SDI.2, FCS_COP.1, FPT_ITT.1/Runtime	Section 4.3.1
O.TEE_ISOLATION	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime	Section 4.3.1
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	Section 4.3.1
O.ROLLBACK_PROTECTION	FDP_SDI.2/Rollback, FPT_FLS.1/Rollback	Section 4.3.1
O.TA_PERSISTENT_TIME	FPT_STM.1/Persistent Time, FMT_MTD.1/Persistent Time, FMT_SMF.1/Persistent Time	Section 4.3.1

Security Objectives	Security Functional Requirements	Rationale
O.DEBUG	FDP_ACC.1/Debug, FDP_ACF.1/Debug, FCS_COP.1/Debug, FMT_SMR.1/Debug, FIA_UID.2/Debug, FIA_ATD.1/Debug, FIA_USB.1/Debug, FIA_UAU.2/Debug, FIA_UAU.6/Debug	Section 4.3.1

Table 11 Security Objectives and SFRs - Coverage

Security Functional Requirements	Security 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
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	O.TEE_ID
FAU_STG.1	O.TEE_ID
FPT_STM.1/Instance time	O.INSTANCE_TIME

Security Functional Requirements	Security Objectives
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
FMT_MSA.3/Trusted Storage	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FDP_ITT.1/Trusted Storage	O.TRUSTED_STORAGE
FDP_SDI.2/Rollback	O.OPERATION, O.ROLLBACK_PROTECTION
FPT_FLS.1/Rollback	O.OPERATION, O.ROLLBACK_PROTECTION
FPT_STM.1/Persistent Time	O.TA_PERSISTENT_TIME
FMT_MTD.1/Persistent Time	O.TA_PERSISTENT_TIME
FMT_SMF.1/Persistent Time	O.TA_PERSISTENT_TIME
FDP_ACC.1/Debug	O.OPERATION, O.DEBUG
FDP_ACF.1/Debug	O.OPERATION, O.DEBUG
FCS_COP.1/Debug	O.DEBUG
FMT_SMR.1/Debug	O.OPERATION, O.DEBUG
FIA_UID.2/Debug	O.OPERATION, O.DEBUG
FIA_ATD.1/Debug	O.OPERATION, O.DEBUG
FIA_USB.1/Debug	O.OPERATION, O.DEBUG
FIA_UAU.2/Debug	O.OPERATION, O.DEBUG
FIA_UAU.6/Debug	O.OPERATION, O.DEBUG

Table 12 SFRs and Security Objectives

7.3.3 Dependencies

7.3.3.1 SFRs Dependencies

Requirements	CC Dependencies	Satisfied Dependencies

Requirements	CC Dependencies	Satisfied Dependencies
FDP_ACC.1/Debug	(FDP_ACF.1)	FDP_ACF.1/Debug
FDP_ACF.1/Debug	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/Debug
FCS_COP.1/Debug	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	
FMT_SMR.1/Debug	(FIA_UID.1)	FIA_UID.2/Debug
FIA_UID.2/Debug	No Dependencies	
FIA_ATD.1/Debug	No Dependencies	
FIA_USB.1/Debug	(FIA_ATD.1)	FIA_ATD.1/Debug
FIA_UAU.2/Debug	(FIA_UID.1)	FIA_UID.2/Debug
FIA_UAU.6/Debug	No Dependencies	
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	(FAU_GEN.1)	
FAU_STG.1	(FAU_GEN.1)	

Requirements	CC Dependencies	Satisfied Dependencies
FPT_STM.1/Instance time	No Dependencies	
FCS_RNG.1	No Dependencies	
FDP_ACC.1/Trusted Storage	(FDP_ACF.1)	FDP_ACF.1/Trusted Storage
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
FDP_SDI.2/Rollback	No Dependencies	
FPT_FLS.1/Rollback	No Dependencies	
FPT_STM.1/Persistent Time	No Dependencies	
FMT_MTD.1/Persistent Time	(FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_SMF.1/Persistent Time
FMT_SMF.1/Persistent Time	No Dependencies	

Table 13 SFRs Dependencies

7.3.3.2 Rationale for the exclusion of Dependencies

The dependency FMT_MSA.3 of FDP_ACF.1/Debug is discarded. There is no management of security attributes by authorized users for this access control SFP as security attributes are either exclusively managed by the TSF or not modifiable during the end-usage phase, therefore the dependency FMT_MSA.3 is not applicable.

The dependency FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2 of FCS_COP.1/Debug is discarded. The TEE Debug authentication key used for authenticating TEE Debug Administrator in FCS_COP.1 is set during manufacturing. It cannot be changed during the end-usage phase.

The dependency FCS_CKM.4 of FCS_COP.1/Debug is discarded. The TEE Debug authentication key used for TEE Debug Administrator authentication in FCS_COP.1/Debug is not required to be changed or destroyed during the end-usage phase.

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

The dependency FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2 of FCS_COP.1 is discarded. The TEE storage root of trust cryptographic key used for cryptographic operations in FCS_COP.1 is set during manufacturing. If a derived key is used for trusted storage, the ST writer will have to add a depency to FCS_CKM.1 and specify the derivation method.

The dependency FCS_CKM.4 of FCS_COP.1 is discarded. The TEE storage root of trust used for cryptographic operations in FCS_COP.1 is not required to be changed or destroyed during the end-usage phase.

The dependency FAU_SAA.1 of FAU_ARP.1 is discarded. The potential security violations are explicitly defined in the FAU_ARP.1 requirement. there is no audited event defined in the SFR of this PP.

The dependency FAU_GEN.1 of FAU_SAR.1 is discarded. This dependency is discarded as the only audit record considered is the TEE identifier and this identifier is set before TOE delivery and non-modifiable afterwards.

The dependency FAU_GEN.1 of FAU_STG.1 is discarded. This dependency is discarded as the only audit record considered is the TEE identifier and this identifier is set before TOE delivery and non-modifiable afterwards.

7.3.3.3 SARs Dependencies

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

Table 14 SARs Dependencies

7.3.4 Rationale for the Security Assurance Requirements

The assurance level defined in this Protection Profile 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 defined in Annex A.

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, 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, stated in Annex A.

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 EAL 2 product recognition for the schemes that do not recognize the AVA_TEE.2 component.

A. Application of Attack Potential to TEE

This Annex introduces the methodology for evaluating the attack potential for Trusted Execution Environment, to be used with the TEE Protection Profile. This work is based on TEE industry stakeholders' experience, gathered within GlobalPlatform TEE Security Working Group and TEE Attacks Expert Working Group.

It provides guidance metrics to calculate the attack potential required by an attacker to perform an attack. It includes the definition of a TEE-specific attack quotation table, and provides example attacks on TEE implementations. The goal is to help evaluating the effort required to perform a successful attack on a TEE. This document is compatible with [CC3] and [CEM].

A.1 Attack Quotation Grid

The attack potential that is necessary to carry out a TEE attack is the result of a combination of five factors in the attack's identification and exploitation phases: Elapsed time, Access to the TOE, Expertise, Knowledge of the TOE, Equipment and Open Samples. The following modifications have been introduced to the smartcards' rating table:

- Access to the TOE ranges have been modified to take into account the actual number of samples (SoC or devices) that may be needed to perform the attacks, which is lower than for smartcard products.
- There are four levels of knowledge of the TOE in the TEE attack rating table: Public, Restricted, Sensitive and Critical, with the following modifications:
    - Restricted knowledge of the TOE requires NDA or similar information control policy;
    - Sensitive knowledge of the TOE requires additional independent developer's site audit;
    - Critical knowledge of the TOE is not reachable under the current TEE Security Evaluation Methodology;
    - Moreover, Very Critical hardware design knowledge of the TOE has been removed, considering that many hardware resources are shared between the TEE and the REE, which results in Critical knowledge already implying the full knowledge of the hardware blocks that the TEE relies on;
- There are four levels of open samples: Public, Restricted, Sensitive and Critical, which can be used under the same conditions specified for the knowledge of the TOE.

defines the values of these factors. It is derived from the standard table defined in CC v3.1 revision 4, and includes modifications very similar to those in CC Supporting Document "Application of Attack Potential to Smartcards" version 2.9 [APSC].

The main modification with respect to the standard CC attack potential table consists in the separation of the attack's identification and exploitation phases. The identification phase corresponds to the initial creation of an attack up to the point where it is ready to be exploited, i.e. until a detailed description and setup, as well as potentially necessary new tools created on purpose, are available. The exploitation phase corresponds to the use of the analysis, techniques and tools defined in the identification phase to perform the attack successfully on each TOE. Note that very often the potential required in the exploitation phase is lower than the one required in the identification phase. This happens, for instance when the shortest path for finding a software vulnerability requires hardware means, while, exploiting such a vulnerability on any other device with the same software is straightforward.

Factor	Identification	Exploitation
Elapsed time		
<= one hour	0	0
<= one day	1	3
<= one week	2	4
<= one month	3	6
> one month	5	8
Not practical	*	*
Access to the TOE		
Only 1 sample	0	0
<10 samples	1	2
<30 samples	2	4
>30 samples	3	6
Not practical	*	*
Expertise		
Layman	0	0
Proficient	2	2
Expert	5	4
Multiple experts	7	6
Knowledge of the TOE		
Public	0	0
Restricted	2	2
Sensitive	4	3
Critical	NA (6)	NA (5)
Equipment		
None	0	0
Standard	1	2
Specialized	3	4
Bespoke / Multiple specialized	5	6
Multiple bespoke	7	8
Open samples		
Public	0	NA
Restricted	2	
Sensitive	4	
Critical	NA (6)	

Table 15: TEE attack potential

The meaning of the factors and the ranges of values are the following:

- Elapsed time
    - This is the time spent in the identification and the exploitation phases. The Elapsed time in the identification phase is the time taken by the steps that are necessary to be performed only once in order to carry out the attack on many TOEs later. In the exploitation phase, it is the time required by the steps that must be performed for every

TOE that is attacked. Elapsed time ranges from man-hours to man-months; for calculation purposes, one day is about 8 hours, one week is about 40 hours and one month, about 160 hours.

- Access to the TOE (target devices)
    - This is the number of samples of the TOE that is necessary in the identification and the exploitation phases. This number ranges from 1 to more than 30.
- Expertise
    - Layman: attacker with no particular expertise, unknowledgeable compared to Experts or Proficient persons;
    - Proficient: knowledgeable in that they are familiar with the security behaviour of the product type;
    - Expert: familiar with the underlying algorithms, protocols, hardware, structures, security behaviour, relevant security principles and concepts, techniques and tools for the definition of new attacks, cryptography, classical attacks for the product type, attack methods, etc. implemented in the product type;
    - Multiple expert: many fields of expertise are required to perform distinct steps of the attack, such as side-channel attacks expertise and software attack expertise.
- Knowledge of the TOE (i.e. knowledge of specifications, design information, guidance documentation, source code)
    - Public: information in the public domain;
    - Restricted: knowledge that is controlled within the developer organization (only accessible to some employees on need-to-know basis) and shared with other organizations under a Non-Disclosure Agreement (NDA). The information security policy of the developer could be considered as appropriate evidence to justify the usage of this level in a real TEE evaluation. No site audit is required as part of the TEE Security Evaluation Methodology;
    - Sensitive: knowledge that is shared between discrete teams within the developer organization (e.g. security team or cryptographic team only), access to which is limited to authorized members of the specified teams. Sensitive knowledge of the TOE requires developer's site audit, which should allow gaining confidence in the security measures applied by the developer and on the actual protection of the assets. Independent site audit against ISO 27001, from CC or EMVCo certification bodies could be accepted to justify the usage of this level in a real TEE evaluation provided the audit covers the TEE assets under evaluation;
    - Critical: knowledge that is known by few individuals only (duly cleared employees who have access to the cryptographic algorithms for instance), access to which is tightly controlled and tracked on a strict need-to-know basis under individual NDA. Critical knowledge would provide 6 and 5 points in identification and exploitation phases, respectively. However, Critical knowledge of the TOE is not reachable under the current TEE Security Evaluation Methodology (third-party site audits are not sufficient to reach this level).
- Equipment
    - Standard: readily available to the attacker, either for the identification of a vulnerability or for an attack. Example of standard equipment include: software tools available or downloadable from the internet, such as software debuggers, protocol analyzers, or

rooting tools exploiting flaws in a PC-device protocol. Hardware equipment is limited to a PC client and cables to connect to the target device;

- Specialized: not readily available to the attacker, but could be acquired without undue effort. Examples include power analysis tools, micro-probe workstation, laser equipment, or use of hundreds of PCs linked across the Internet. Specialized equipment may be revised to Standard in case equipment becomes available to buy from the Internet at a reasonable cost;
- Bespoke: not readily available to the public as it may need to be specially produced, or because the equipment is so Specialized that its distribution is controlled, possibly even restricted. Alternatively, the equipment may be very expensive, or consist of several Specialized equipment units;
- Multiple bespoke: different types of Bespoke equipment are required for distinct steps of an attack.

The following table shall be used in the equipment rating:

Equipment	**Classification**
UV-light emitter	Standard
Flash light	Standard
Low-end visible-light microscope	Standard
Climate chamber	Standard
Voltage supply	Standard
Analogue oscilloscope	Standard
PC or work station	Standard
Signal analysis software	Standard
Signal generation software	Standard
High-end visible-light microscope and camera	Specialized
UV light microscope and camera	Specialized
Micro-probe Workstation	Specialized
Laser equipment[1]	Specialized
Signal and function processor	Specialized
High-end digital oscilloscope	Specialized
Signal analyzer	Specialized
Tools for chemical etching (wet)[2]	Specialized
Tools for chemical etching (plasma)	Specialized
Tools for grinding	Specialized
Design verification and failure analysis tools	
Scanning electron microscope (SEM)	Bespoke
E-beam tester	Bespoke

[1] Off-the-shelf laser equipment is Specialized. Laser setups that contain many proprietary components or several Specialized components may become Bespoke.

[2] Chemical etching may be performed without Specialized equipment with Standard chemical tools found in drugstores. However a dedicated machine may be classified as Specialized.

Atomic Force Microscope (AFM)	Bespoke
Focused Ion Beam (FIB)	Bespoke
New Tech Design Verification and Failure Analysis Tools	Bespoke
Software tools	
Software tools[3]	Standard

Table 16: Equipment rating table for TEE attacks

- Open samples

  A TEE open sample is a TEE on which an attacker can install and execute TA code. A TEE with known secrets is a TEE for which the attacker knows essential secrets such as private keys necessary to sign and install Trusted Applications. This factor is not meaningful in the exploitation phase. There are four classes of open samples:

  - Public: Public open samples mean that devices/platforms providing an open TOE (TEE), i.e. either without limitations on the installation of Trusted Application, so that anyone may compile and install any Trusted Application on the device/platform or with privileged access to the TOE are publicly available (e.g. to anyone asking, without NDA nor control of delivery). For samples with known secrets, the Public level means that the secrets are readily available by anyone having Public knowledge of the TOE;
  - Restricted: Restricted open samples mean that open samples are accessible with a level of control equivalent to the control used for providing Restricted knowledge of the TOE, such as detailed datasheets. For samples with known secrets, this level applies to the situation where the secrets are delivered with some control to requesters, e.g. under NDA. The information security policy of the developer could be considered as appropriate evidence to justify the usage of this level in a real TEE evaluation. No site audit is required as part of the current TEE Security Evaluation Methodology;
  - Sensitive: Sensitive open samples mean that open samples are accessible with a level of control equivalent to the control used for providing Sensitive knowledge of the TOE, such as low-level design documentation. For samples with known secrets, this means that the secrets are delivered under NDA with strong control and tracking of the samples on which these secrets can be used, with the assurance that the receiving organization is able to setup a control at the same level. Sensitive level requires developer's site audit, which should allow gaining confidence in the security measures applied by the developer and on the actual protection of the open samples. Independent site audit against ISO 27001 or from CC or EMVCo certification bodies could be accepted to justify the usage of this level in a real TEE evaluation provided the audit covers the TEE open samples;
  - Critical: Critical open samples mean that they are accessible with a level of control equivalent to the control used for providing Critical knowledge of the TOE, such as the full source code, VHDL or hardware layout. Usage of this level also implies that very few open samples are produced and their usage is tracked very thoroughly. For samples with known secrets, this means that the secrets are only known by only a few individuals and are not delivered. Critical level would provide 6 points in the

[3] All software tools are Standard. Specifically developped tools or extensions should be rated through the expertise and time parameters.

identification phase. However, this level is not reachable under the current TEE Security Evaluation Methodology (third-party site audits are not sufficient).

In the ecosystem of devices implementing a TEE, open samples availability is often Restricted: SoC vendors usually implement some level of control on full devices, and as manufacturers or operators, further in the value chain, may have the ownership of the TEE, the typical situation would be that open samples are available only under NDA, but without further restrictions as open samples have to be delivered to such stakeholders.

The following modifications have been introduced to the smartcards' rating table:

- Access to the TOE ranges have been modified to take into account the actual number of samples (SoC or devices) that may be needed to perform the attacks, which is lower than for smartcard products.
- There are four levels of knowledge of the TOE in the TEE attack rating table: Public, Restricted, Sensitive and Critical, with the following modifications:
  - Restricted knowledge of the TOE requires NDA or similar information control policy;
  - Sensitive knowledge of the TOE requires additional independent developer's site audit;
  - Critical knowledge of the TOE is not reachable under the current TEE Security Evaluation Methodology;
  - Moreover, Very Critical hardware design knowledge of the TOE has been removed, considering that many hardware resources are shared between the TEE and the REE, which results in Critical knowledge already implying the full knowledge of the hardware blocks that the TEE relies on;
- There are four levels of open samples: Public, Restricted, Sensitive and Critical, which can be used under the same conditions specified for the knowledge of the TOE.

The attack potential required by a TEE attack path is the sum of the factors of the identification and exploitation phases. This Protection Profile requires resistance to attackers with TEE-Low attack potential, which means that the TEE must resist to attacks below 21 points. Table 17 provides the hierarchy of attack potential applicable to the TEE. It plays the same role as the table defined in CC v3.1 and the smartcard table defined in [APSC].

Attack potential values	Attack potential required to exploit the scenario:	TOE resistant to attackers with attack potential of:
0-15	TEE-Basic	No rating
16-20	TEE-Low	TEE-Basic
21-24	TEE-Moderate	TEE-Low
25-30	TEE-High	TEE-Moderate
31 and above	Beyond TEE-High	TEE-High

Table 17: Rating of TEE resistance

A.2 Attackers' Exploitation Profiles

This section describes the profile of the attackers that the GlobalPlatform Trusted Execution Environment is expected to protect against. The general goal of the TEE is to prevent attacks from being widespread through the Internet or other means to many end-users at a minimal cost. Hence, we consider different attacker profiles for the identification and for the exploitation phases.

In the identification phase, a typical attacker will not hesitate to spend time and effort, and to use non-standard software or hardware means, in order to find a vulnerability that can be further exploited through easier means, so that it can be widespread by, for instance, making a software exploiting the vulnerability available on the Internet. We therefore do not arbitrarily limit the attack paths used for identification.

The exploitation of the attack will typically not be performed directly/locally by the attacker who has performed the identification. Instead, it will either be performed remotely – by the original attacker or, very often, by another entity interested in making the attack spread more widely – without end-user awareness, or locally on behalf of the end-user. We define several example exploitation profiles. Likely the attacker of the exploitation phase has much more limited resources than the attacker performing the identification, and the exploits he performs should require only software means and Standard equipment. Moreover, only attacks with non-destructive exploitation are considered realistic in the TEE threat model, as end-users usually consider that the risk of breaking their devices outweighs any possible benefit from the attacks. However, when the secret under attack is TOE-specific and cannot be used against other instances of the TOE, the identification must be followed by an exploitation phase that uses the same amount of resources.

The list of profiles includes a profile for remote exploitation/malware, and three local exploitation profiles, where the exploitation is carried out locally at the initiative of the end-user on his device. Note that Profile 4 can perform hardware exploitation; such an exploitation will often lead to an attack rating of 21 points or higher; therefore, Profile 4 is often beyond reach.

Additional exploitation profiles may be used for any attack listed in this document or defined in the framework of an actual TEE evaluation.

Factor	Value	Exploitation Profile			
		1 Remote	2 Local Layman	3 Local Proficient	4 Local Proficient with Equipment
Elapsed time					
<= one hour	0		0		
<= one day	3			3	
<= one week	4	4			4
<= one month	6				
> one month	8				
Not practical	*				
Access to TOE					
Only 1 sample	0	0	0	0	0
<10 samples	2				
<30 samples	4				
>30 samples	6				
Not practical	*				
Expertise					
Layman	0		0		
Proficient	2	2		2	2
Expert	4				
Multiple experts	6				
Knowledge of the TOE					
Public	0	0	0	0	0
Restricted	2				
Sensitive	3				
Critical	NA(5)				
Equipment					
None	0				
Standard	2	2	2	2	
Specialized	4				4
Bespoke / Multiple specialized	6				
Multiple bespoke	8				
Open samples					
Public	NA				
Restricted					
Sensitive					
Critical					
Total		**8**	**2**	**7**	**10**

Table 18: Ratings of Exploitation Profiles 1 to 4

A.2.1 Exploitation Profile 1 (Remote Attacker)

This exploitation profile corresponds to performing the attack on a remotely-controlled device or alternatively making a local tool that is so convenient that usual end-users will be easily convinced to download and use it. The attacker, if different from the one that performed the identification, retrieves the details on the vulnerability identified in the identification phase and minimal outputs such as local 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 very easily usable tool available on the Internet (one-stop-shop). The attacker needs a Proficient level of expertise to do such a tool, and it will take around one week to design it provided that there are similar tools/programs available on the Internet, which is almost always the case. The attack requires Standard equipment consisting of source code available from the Internet that he will use as the code base for his own application or tool. This is standard practice to reuse an existing base when designing a new malware, Trojan, virus, or rooting tool etc. No specific equipment from the end-user is needed.

A.2.2 Exploitation Profile 2 (Local Layman Attacker)

This exploitation profile corresponds to performing the attack on a local device, requiring physical access to the target device, with the end-user performing the attack. The attacker retrieves example attack code/application from the identifier, guidelines written on the Internet on how to perform the attack, requiring downloading and using tools to jailbreak/root/reflash the device in order to get privileged access to the REE allowing the execution of the exploit. The attacker does not need any specific level of expertise as he is following existing guidelines that someone will have posted, and possibly enhanced, on the Internet. The attack requires Standard equipment consisting of tools available from the Internet that the attacker will use to perform the attack, as well as a PC connected to the device. The attack will typically take less than one hour.

A.2.3 Exploitation Profile 3 (Local Proficient Attacker)

This exploitation profile corresponds to performing the attack on a local device, requiring physical access to the target device and Proficient level of expertise. Even though there exist end-users with this level of expertise, the assumption for exploitation, in order to be spread very largely, is that the general local attacker for exploitation does not, which means that the attack is likely performed by an attacker on behalf of the user. The exploitation quotation is the same as exploitation profile 2, except from the level of expertise (Proficient) and the Elapsed time (less than one day, but more than one hour, to take into account external intervention).

A.2.4 Exploitation Profile 4 (Local Proficient Attacker with Equipment)

This exploitation profile corresponds to performing the attack on a local device, requiring physical access to the target device, with the end-user having a specialized entity (backstreet shop) to perform the attack instead of him, with a Proficient level of expertise. Compared with exploitation profile 2, this attack requires a higher level of expertise (Proficient), equipment (Specialized) and takes longer (one week, which also materializes the fact that the end-user has to bring in a device and get it back later).

A.3 Examples of Attack Paths

This section provides examples of attack paths using hardware and software means that may lead to successful identification phases in specific TEE implementations. The exploitation phase is not described here (cf. Exploitation Profiles defined in A.2).

The evaluator must consider such kinds of identification attack paths and extend this list as required by the characteristics of the TOE.

A.3.1 Hardware-based Attack Paths

The purpose of this section is to describe several attack paths relying on hardware means that may lead to successful identification phases.

A.3.1.1 Side Channel Analysis Attack

The goal of the attack presented here is to retrieve a cryptographic key used in the Trusted Storage for protecting an asset belonging to a specific Trusted Application. The cryptographic key is retrieved by breaking the cryptographic protections used in the Trusted Storage implementation, for instance, using power or electromagnetic analysis of the TEE platform during the Trusted Storage operations. Typical requirements for the identification phase include:

- ability to perform cryptographic operations with the targeted key;
- thousands of measurements of cryptographic operations performed by the Trusted Storage with the targeted key;
- availability of open samples in order to perform the sampling efficiently to retrieve the secrets that the Trusted Storage implementation is based on;
- knowledge of the cryptographic schemes used for Trusted Storage in the TEE;
- controlling the REE in order to minimize the noise due to REE processing;
- equipment to measure and analyze power consumption or electromagnetic radiation from outside the SoC or packaged device, and a wire coil or an antenna located close to the device.

Note: All the cryptographic operations which are in the scope of the evaluation (cf. FCS_COP.1 requirements in this Protection Profile), either used internally by the TEE OS or provided to the TAs through an API, are potential targets for side-channel analysis.

A.3.1.2 Fault Injection Attack

The goal of fault injection attacks is to corrupt temporarily the behavior of the TOE using physical means such as a laser pulse, an EM pulse or a glitch on input signals (clock, power, reset). This kind of corruption can alter the execution flow of code or change the interpretation of processed data by the SoC at the benefit of the attacker.

Fault injection attacks can be used to bypass security checks that are implemented in software in the TOE such as signature verification or anti-rollback checks, but can also target hardware accelerators and be used to extract cryptographic keys from them using post analysis over the faulty results (for example Differential Fault Analysis).

Typical requirements for the identification phase include:

- side channel analysis to identify the TOE weakness (no code is required);

- setup of a fault injection attack bench (e.g. digital oscilloscope, pulse generator, fault injection means):
    - preparation of the device to get physical access to the targeted component (SoC or external RAM);
    - control the REE in order to minimize the noise due to REE processing and to control execution within the TEE and to obtain a trigger signal.

A.3.1.3 External DRAM Probing

The goal of this attack is to retrieve the value of a cryptographic key during an AES computation using the TEE OS cryptographic library. With the assumption that the RAM is not part of the package and the plaintext is known, a typical identification phase requires:

- analyzing a TOE sample in order to find out the RAM bus data rate;
- an analyzer supporting the RAM data rate;
- probing the bus in order to localize plaintext data manipulated during the AES computation;
- performing correlation analysis of the plaintext manipulation in order to retrieve the key.

A.3.1.4 Unprotected Debug Interface

The goal of this attack is to directly access and read or modify TEE memory contents by means of a hardware debugging facility, and to use this privilege to find a vulnerability that is exploitable by software as in the external DRAM probing attack, or to modify a value during execution that will end up in performing a privileged action without proper authorization.

Typical requirements for the identification phase includes

- analysis of the system that the JTAG provides access to;
- analysis of the TEE software;
- either finding a vulnerability and designing the related exploit, or directly modifying TEE memory contents to change for instance persistent values without authorization.

A.3.2 Software-based Attack Paths

The purpose of this section is to describe several attack paths relying on software means that may lead to successful identification phases. Note that some of the descriptions are not proper attacks but ways to discover potential vulnerabilities, for instance fuzzing and use of obsolete or hidden APIs.

A.3.2.1 Cache Attack on Crypto

The goal of this attack is to retrieve keys used by the TEE for various operations in a setup where all these conditions are met:

- TEE and REE share the same cache memory;
- the cryptographic algorithm is implemented fully in software;
- the cryptographic algorithm relies on memory accesses, for instance lookup tables.

This attack is carried out by tightly controlling the cache memory content belonging to the REE, allowing to get information on the amount of cache memory allocated to the TEE and measuring TEE cryptography execution times in order to deduce statistics on cache misses and to get information on the cryptographic keys that are used, and finally exploiting the keys to perform unauthorized operations.

Typical requirements for the identification phase include:

- root access to the REE;
- comprehensive TEE documentation including the hardware resources that are used;
- REE debugging tools;
- open samples to be able to load a test TA which invokes the TEE cryptographic operation to sample the target;
- the ability to perform cryptographic operations with the targeted key;
- measurements (e.g. few thousands) by the REE of cryptographic operations performed by by the TEE.

The software used in this attack can ensure that the state of execution on the TEE side is the same at each execution, by restarting the TEE and/or performing cache line eviction and cache flushing so that, for instance:

- all the cache memory belongs to the TEE and nothing is cached at the beginning of execution of the algorithm under scrutiny.

A.3.2.2 Fuzzing on the Client API or the TEE Driver

The goal of this attack is to trigger unexpected behavior and to find vulnerabilities in the TEE implementation by reaching unexpected internal states, using the TEE Client API [1] or the TEE driver interfaces with invalid, unexpected, or random data.

Typical requirements for the identification phase include:

- privileged access to the REE, possibly remote;
- open samples in order to implement/run custom TAs;
- time and resources to implement the fuzzing tool or to adapt existing software to the TEE;
- good knowledge of the TOE in order to evaluate the success of the attack - e.g. by including in the buffer overflow data a binary code that should write to Rich OS memory, and in order to restart the attack upon crash.

A.3.2.3 Breach of Memory Isolation

The goal of this attack is to directly access and modify TEE memory contents by exploiting a bug in the hardware controlling memory isolation.

Typical requirements for the identification phase include:

- Gain privilege to bypass or modify access control or isolation rules to be able to write directly to physical memory;
- TEE documentation with memory mapping or reverse-engineering.

Once the TEE memory is accessed, a software analyzer can be used to reverse-engineer the TEE code and data in a straightforward manner in order to determine what to write to TEE memory to modify TEE persistent data without authorization.

A.3.2.4 Certificate Parsing Error

The goal of this attack is to provide the TEE with a malformed certificate that will be parsed incorrectly, allowing the attacker to inject rogue code inside the secure environment. The attack might target the TEE's update capability, the TA provisioning capability or some implementation-dependent client authentication capability.

It is expected that potential vulnerabilities would be discovered through fuzzing or targeted trials of edge cases, thus not requiring any preliminary knowkedge of the TOE.

A.3.2.5 Use of APIs/Protocols with known vulnerabilities or diagnostic APIs

The goal of this attack is to use a deprecated or undocumented interface to inject malicious code or to extract confidential data into or out of a TEE. The attack vector may be, for example, a legacy communication protocol relying on an insecure cryptographic algorithm, or a proprietary API which has not been kept up-to-date with regard to security fixes.

It is expected that a vulnerability would be discovered in a proprietary API or through inspection of existing applications that use undocumented functionality. Once a potential vulnerability has been identified, the attacker still has to find a way to exploit it. Typical requirements for the identification phase include access to debugging equipment that allows to locate an actual vulnerability and ability to design an exploit for it.