Security Target

Kinibi v311A Security Target

PREFACE

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

VERSION HISTORY

Version	Date	Modification
1.0	2016-04-04	First version
1.1	2016-04-26	Reworked assumptions
1.2	2016-05-03	Completed SFR adaptation; TEE/TOE distinction
1.3	2016-05-24	Feeback from Thales; removed persistent time
1.4	2016-06-01	Updated to PP 1.2; aligned most SFR to the PP
1.5	2016-06-07	Mention TA management (out of scope); reference crypto algorithms document
1.6	2016-08-29	Added the file name of the ATE cover sheet. Removed obsolete reference [ALP]. Added the exact version string to the TOE description.
1.7	2016-11-22	Lifecycle: note that phase 0 is out of scope. TOE overview: mention all embedded TAs, and TAs outside the TOE. List all known TAs/drivers and their versions.
	2016-12-12	Removed confidentiality notice (no content change)
1.8	2017-01-10	Corrected PP/ST comparison: OE.TEE_FIRMWARE_UPGRADE is not included

1 INTRODUCTION

This document is the Security Target for Kinibi which is a Trustonic implementation of an operating system for a Trusted Execution Environment (TEE), hereafter known as "Trusted Operating System" (Trusted OS).

A TEE is a secure, integrity-protected processing environment, consisting of processing, memory and storage capabilities. It is isolated from the "normal" processing environment, sometimes called the rich execution environment (REE), where the device operating system and applications run.

The TOE addressed by the current ST is a TEE operating system for mobile devices that enable mobile security services such as:

- Digital rights management,
- Online and mobile banking and Payment,
- Enterprise and web-based services
- Content protection

The TOE in the scope of this ST implements the core functionalities defined in GlobalPlatform TEE Internal API Specification [GPIAPI] and Trustonic's proprietary APIs defined in [TR-DEVAPI].

The TOE is defined as the Kinibi trusted execution environment operating system, running on a hardware/Firmware. The firmware as well as the hardware is not part of the TOE. Hardware/Firmware is considered to be part of the TOE environment.

This Security Target describes:

- The Target of Evaluation (TOE)
- The assets to be protected, the threats (T) to be countered by the TOE itself during the usage of the TOE,
- The organizational security policies (OSP), and the assumptions (A),
- The security objectives (OT) for the TOE and its environment (OE),
- The security functional requirements (SFR) for the TOE and its IT environment,
- The TOE security assurance requirements (SAR),
- The TOE Summary specification (TSS).

1.1 ST Identification

Title	Kinibi Security Target
Reference	TT-CC-2016-ST
Version	**1.8**
Date of Issue	**10-01-2017**
ITSEF	Thales
Certification Body	ANSSI
Author	TRUSTONIC
CC Version	3.1 Revision 4
Assurance Level	EAL2 augmented with AVA_TEE.2 which refines the AVA_VAN.2 Basic attack potential to a TEE-Low attack potential
Protection Profiles	The current ST does not claim conformance to any Protection Profile
Status	Final

1.2 TOE Reference

Developer	Trustonic
TOE commercial name	Kinibi
TOE version number	t-base-EXYNOS64-Android-311A-V004-20160527_225213_11082_38854
Chipset Identifiers	Samsung LSI « Joshua » Generic ARMv8
Reference of Chipset	Exynos7870
Version of Chipset	EVT0_REV0.0

Note that the TOE consists of software only. The chipset is out of scope of the evaluation.

1.3 Diffusion List

- Trustonic employees
- Trusted Labs employees
- Thales employees
- ANSSI
- General publication once the evaluation is completed

1.4 Assurance Measures Documents

The following TRUSTONIC's technical reports describe the assurance measures of the TOE:

[ATE]	ATE.txt
[DEL]	Trustonic-Kinibi-ALC_DEL.docx
[CMC]	Trustonic-Kinibi-ALC_CMC_CMS.docx
[CMS]	Trustonic-Kinibi-ALC_CMC_CMS.docx
[FSP]	cc_adv_pre.pdf
[TDS]	cc_adv_tds.pdf
[OPE]	Trustonic-Kinibi-AGD_OPE.docx
[PRE]	Trustonic-Kinibi-AGD_PRE.docx
[ARC]	cc_adv_arc.pdf

2 TECHNICAL TERMS, ABBREVIATION AND ASSOCIATED REFERENCES

2.1 Technical terms

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 storage data which is used at starup is a one for which runtime consistency was enforced on the same previous storage state.* *Consistency implies runtime integrity of what is successfully written and read back – values or code. However the storage data that is used at startup may not come from the latest power cycle. This notion is then 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	*Monotony is the property of 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.*

Term	Definition
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.*
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 inatance 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 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	*An 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.*

2.2 Abbreviation

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])*
EE	*Execution Environment*
ID	*IDentifier*
FIFO	*First In, First Out*
HD	*HD*
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*

AES	*Advanced Encryption Standard (defined in [AES])*
API	*Application Programming Interface*
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	*RFC*
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])*
TSF	*TOE Security Functionality (defined in [CC1])*
TSFI	*TSF Interface (defined in [CC1])*
TSS	*TOE Summary Specification*
USB	*Universal Serial Bus*
VPN	*VPN*

2.3 Associated references

[AES]	ADVANCED ENCRYPTION STANDARD (AES). FIPS PUB 197. November 2011.
[CC1]	Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1. Revision 4. September 2012. CCMB-2012-09-001.
[CC2]	Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1. Revision 4. September 2012. CCMB-2012-09-002.

[CC3]	Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements. Version 3.1. Revision 4. September 2012. CCMB-2012-09-003.
[CEM]	Common Methodology for Information Technology Security Evaluation, Evaluation Methodology. Version 3.1. Revision 4. September 2012. CCMB-2012-09-004.
[COMP]	Common Criteria mandatory technical document – Composite product evaluation for smart cards and similar devices, CCDB-2012-04-001, Version 1.2, April 2012.
[DES]	DATA ENCRYPTION STANDARD (DES). FIPS PUB 46-3. October 1999.
[GPCAPI]	TEE Client API Specification version 1.0, GlobalPlatform, July 2010.
[GPIAPI]	TEE Internal Core API Specification version 1.1, GlobalPlatform, June 2014
[JTAG]	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
[OMTP-TR1]	Open Mobile Terminal Platform Advanced Trusted Environment OMTP TR1 v1.1
[PP-TEE]	GlobalPlatform Device Committee - TEE Protection Profile Version 1.2, Public Release November 2014, Reference: GPD_SPE_021
[RSA]	RSA Cryptographic Standard. PCKS#1 v2.2. October 2012
[SA]	TEE System Architecture version 1.0, GlobalPlatform, December 2011.
[SHA]	SECURE HASH STANDARD. FIPS PUB 180-4. March 2012
[TR-CAK]	Trustonic – Cryptographic Algorithms in Kinibi
[TR-DEVAPI]	Trustonic – Kinibi Developer API
[TR-DEVGDE]	Trustonic – Kinibi Developer Guide
[TR-DRVAPI]	Trustonic – Kinibi Driver Developer API
[TR-DRVGDE]	Trustonic – Kinibi Driver Developer Guide
[TR-KSA]	Trustonic – Kinibi Security Architecture
[WP]	The Trusted Execution Environment: Delivering Enhanced Security at a Lower Cost to the Mobile Market, GlobalPlatform White paper, Feb 2011
[WP-JEE14]	Kinibi – a Trusted Execution Environment. Trustonic White paper, 2014-2016

3 CONFORMANCE CLAIMS

3.1 CC Conformance

This Security Target claims conformance to the following documents defining the ISO/IEC 15408:2005 standard:

- Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, CCMB-2012-09-001, Version 3.1, Revision 4, September 2012 [CC1].
- Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Requirements, CCMB-2012-09-002, Version 3.1, Revision 4, September 2012 [CC2].
- Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements, CCMB-2012-09-003, Version 3.1, Revision 4, September 2012 [CC3].
- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, CCMB-2012-09-004, Version 3.1, Revision 4, September 2012 [CEM].

Conformance to ISO/IEC 15408:2005 is claimed as follows:

- Part 1: conformant
- Part 2: extended with
  - FCS_RNG Random numbers generation
  - FPT_INI.1 TSF Initialisation

  All the other security requirements have been drawn from the catalogue of requirements in [CC2].
- Part 3: EAL2 (augmented with AVA_TEE.2).

3.2 PP Claims

This Security Target is built using items from [PP-TEE].

3.3 PP TEE vs. Kinibi ST

The Kinibi TOE differs from the TOE defined in [PP-TEE] that contains SW/HW/FW parts: the Kinibi TOE is limited to the Trusted OS. This focuses on the security requirements for Kinibi, which constitutes the TOE; the HW part is considered as the environment of Kinibi, covered by security objectives for the environment. This difference causes changes in the current ST regarding the TOE in [PP-TEE]. These changes affect:

1. SFRs
   a. SFRs that are fulfilled by the Chipset should disappear from the current ST.
   b. SFRs that are fulfilled by both SW and HW parts should be modified in the current ST to cover only the SW part.
2. New Assumptions or Objectives for environment will be added to environmentally protect the TOE and cover the security of HW/FW parts.
3. Objectives for the TOE (O)

a. Objectives for the TOE in the PP TEE may become Objectives for the environment in the ST.

The following table shows the SPD of the [PP-TEE] that are applicable to the Kinibi ST:

TOE SPDs	PP TEE	Included
Assumptions		
A.PROTECTION_AFTER_DELIVERY	×	×
A.ROLLBACK	×	×
A.TA_DEVELOPMENT	×	×
A.Configuration		×
A.CONNECT		×
A.PEER.FUNC		×
A.PEER.MGT		×
A.Power-Up		×
A.RNG		×
Threats		
T.ABUSE_FUNCT	×	×
T.CLONE	×	×
T.FLASH_DUMP	×	×
T.IMPERSONATION	×	×
T.ROGUE_CODE_EXECUTION	×	×
T.PERTURBATION	×	×
T.RAM	×	×
T.RNG	×	×
T.SPY	×	×
T.TEE_FIRMWARE_ROLLBACK	×	
T.STORAGE_CORRUPTION	×	×
Organizational Security Policies		
OSP.TEE_ID	×	×

TOE SPDs	PP TEE	Included
OSP.INTEGRATION_CONFIGURATION	×	×
OSP.SECRETS	×	×
OSP.TEE_FIRMWARE_UPGRADE	×	

Table 1 PP SPDs vs. ST

The following table shows the security objectives of the [PP-TEE] that are applicable to the Kinibi ST:

TOE Objectives	PP TEE	Included
Security Objectives for the TOE		
O.CA_TA_IDENTIFICATION	×	×
O.KEYS_USAGE	×	
O.TEE_ID	×	×
O.INITIALIZATION	×	changed into OE. INITIALIZATION
O.INSTANCE_TIME	×	x
O.OPERATION	×	×
O.RNG	×	×
O.RUNTIME_CONFIDENTIALITY	×	×
O.RUNTIME_INTEGRITY	×	×
O.TA_AUTHENTICITY	×	×
O.TA_ISOLATION	×	×
O.TEE_DATA_PROTECTION	×	×
O.TEE_FIRMWARE_UPGRADE	×	
O.TEE_ISOLATION	×	×
O.TRUSTED_STORAGE	×	×
Security Objectives for the Operational Environment		
OE.INTEGRATION_CONFIGURATION	×	×

TOE Objectives	PP TEE	Included
OE.PROTECTION_AFTER_DELIVERY	×	×
OE.ROLLBACK	×	×
OE.SECRETS	×	×
OE.TA_DEVELOPMENT	×	×
OE.TEE_FIRMWARE_UPGRADE	×	
OE.UNIQUE_TEE_ID	×	×
OE.Configuration		×
OE.TRUSTED_HARDWARE		×
OE.TRUSTED_FIRMWARE		×
OE.TA_MANAGEMENT		×
OE.RNG		×
OE.INITIALIZATION		×

Table 2 PP Security Objectives vs. ST

The following table shows the SFRs of the [PP-TEE] that are applicable to Kinibi functionality.

TOE SFRs (in PP TEE)	Kinibi (SFRs included in this ST)	Hardware/Firmware
FIA_ATD.1	amended	no CA identity, only TA identity and CA/TA distinction
FIA_UID.2	×	
FIA_USB.1	×	
FMT_SMR.1	×	
FDP_IFC.2/Runtime	×	shared between SW and HW: relies on correct MMU configuration (in scope) + correct MMU operation (assumption A.PEER.FUNC)
FDP_IFF.1/Runtime	×	same remark as FDP_IFC.2/Runtime
FDP_ITT.1/Runtime		hardware only
FDP_RIP.1/Runtime	×	
FPT_ITT.1/Runtime	×	
FCS_COP.1	×	
FDP_ACC.1/TA_keys		Key usage restrictions are not enforced
FDP_ACF.1/TA_keys		Key usage restrictions are not enforced
FMT_MSA.1/TA_keys		Key usage restrictions are not enforced
FMT_MSA.3/TA_keys		Key usage restrictions are not enforced
FAU_ARP.1	×	shared responsibility, e.g. aborts on invalid memory accesses → hw; integrity verifications → TOE
FDP_SDI.2	×	
FPT_FLS.1	×	shared between SW and HW

TOE SFRs (in PP TEE)	Kinibi (SFRs included in this ST)	Hardware/Firmware
FPT_INI.1	modified	shared responsibility (the TOE is loaded by a firmware bootloader)
FMT_SMF.1	×	
FTP_TEE.1	×	
FAU_SAR.1	×	shared responsibility between SW and HW: retrieved or computed in the TOE from hw-protected data
FAU_STG.1	×	shared responsibility (hw configured by sw)
FPT_STM.1/Instance time	×	
FCS_RNG.1	×	combination: hardware entropy source + crypto PRNG
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	×	
Additional SFRs		
	FCS_CKM.1	Based on hardware random (assumption A.RNG)
	FCS_CKM.4	

Table 3 PP SFRs vs. ST

4 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.

4.1 TOE Type and Definition

The TOE is the Kinibi software layer that can be integrated on different System on Chip (SoC) supporting the ARM TrustZone technology. The TOE defined in this ST contains only the SW part developed by TRUSTONIC and selects elements from [PP-TEE] that are applicable only to Kinibi (Table 3). Integrators may add additional firmware and drivers in the TEE; those are not part of the TOE. Trusted applications are not part of the TOE.

The TEE software architecture identifies three distinct classes of components:

- The Trusted Applications that run on Kinibi and use parts of the GP TEE Internal API [GPIAPI] and the proprietary Kinibi APIs [TR-DEVAPI]
- The Trusted Drivers that run on Kinibi and use the proprietary Kinibi Driver APIs [TR-DRVAPI]
- 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 GP TEE Internal API [GPIAPI] and a proprietary Kinibi Internal APIs [TR-DEVAPI].

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

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

The TOE is a trusted 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 Trusted Drivers (TD) and provides them with a comprehensive set of security features including:

- Functional runtime environment with portable APIs,
- Trusted application separation through the security focused microkernel and
- Protection of sensitive assets through access control and cryptography

Kinibi uses ARM TrustZone to separate the platform into two distinct areas, the Normal World with a conventional rich operation system and rich applications and the Secure World.

The TOE comprises:

- All IT security functionality necessary to ensure the secure execution of Kinibi
- The guidance for the secure usage of Kinibi OS after delivery.

The TOE does not comprise:

- The Trusted Applications (TA) that are not embedded into Kinibi image
- The Trusted Drivers that are not embedded into Kinibi image
- The Rich Execution Environment
- The Client Applications CA
- The Firmware/Hardware device platform.

In the following, TOE and Kinibi are used interchangeably.

4.2 Kinibi SW Architecture

The diagram in Figure 1 shows the TOE in the framework of a hardware device.

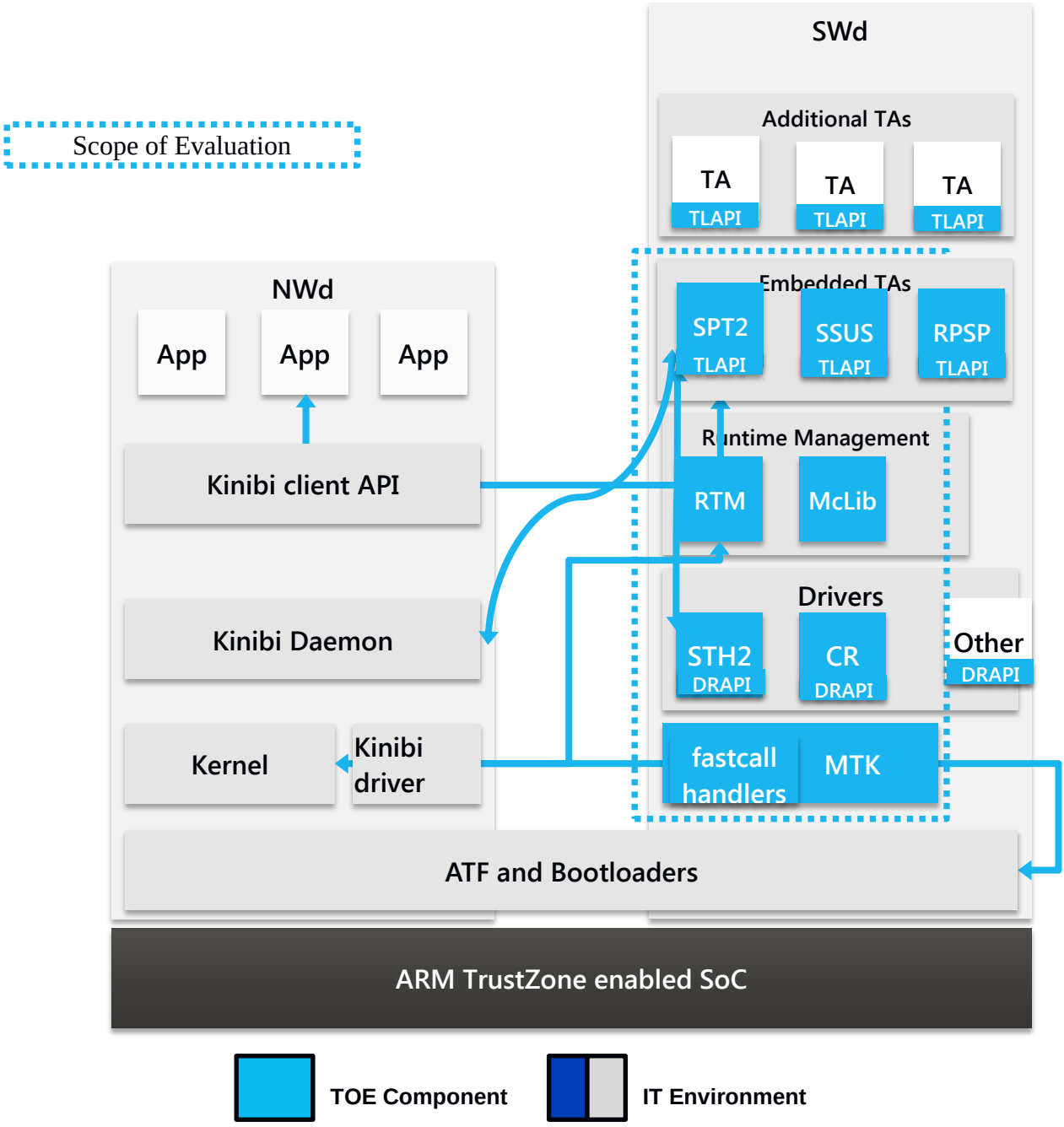

Figure 1: Illustration of Kinibi components

Secure World components contain:

- Kinibi itself, i.e. the secure operating system running in Secure World, comprising:
  - The kernel MTK
  - The runtime manager RTM
  - The Crypto driver CR
  - The Secure Filesystem Driver STH2
  - Additional embedded trusted applications:

        - the secure storage upgrade service (SSUS)
        - the storage proxy (SPT2)
        - the rollback-protected storage proxy (RPSP)
    - A shared code library McLib, whose code is executed in the isolation space of its callers.
- Additional Trusted Applications provided by Trustonic. Such components are out of scope of the TOE.
    - Content Management Trusted Application (version 3.6) (System TA responsible for Containers management).
    - Android Gatekeeper and Keymaster System TAs (version M-MR1), provided by Trustonic, if installed by the OEM.
- The Trusted User Interface driver (co-written by Trustonic, SiP and OEM) (version 311A). This component is out of the scope of the TOE.
- Additional Trusted Drivers and Trusted Applications (System TAs) provided by the integrator (SiP or OEM). Such components are out of scope of the TOE.
- Additional trusted applications installed via Content Management (Service Provider TAs and Installed TAs). Such components are out of scope of the TOE.

The TOE components are as follows (Table 4):

Kinibi Component	Descriptions
MTK	Microkernel running in the TrustZone secure world, which provides isolation between tasks, interprocess communication, and preemptive scheduling. In addition, the kernel hosts fastcall handlers for message exchange with t he REE and architecture and porting layers to interface with the hardware.
RTM	Task in Kinibi which handles memory management, session management, t ask loading, message passing and exception handling.
McLib	Code library that can be used by TAs and drivers in Kinibi. Includes the Kini bi proprietary Internal API and a subset of the GP TEE Internal API.
TLAPI	Set of stubs to call McLib functions from trusted applications.
DRAPI	Set of stubs to call McLib functions from trusted drivers.
CR	Crypto driver, providing cryptographic services to TAs and to other drivers. CR runs as a separate task with driver privileges.

Kinibi Component	Descriptions
STH2	Storage driver, providing secure storage services to TAs and to other drivers. STH2 runs as a separate task with driver privileges.
SSUS	Secure Storage Upgrade Service, to convert secure storage from previous versions of Kinibi. Upgrade is out of scope of the present Security Target.
SPT2	Secure Storage Proxy, a communication relay between the normal world and STH2.
RPSP	Rollback-protected storage proxy, a communication relay between STH2 and a rollback-protected storage driver (which is out of scope of the present Security Target).

Table 4 TOE Architecture

4.3 TOE Environment: Required Hardware/Firmware/Software

The Non-TOE hardware/firmware which allows the installation of the Kinibi on the following hardware systems:

- Samsung LSI Exynos7870, an ARM-based System-on-Chip
- ROM, for secure boot
- Secure write-once memory, e.g. e-fuse for unique SoC identity
- RAM
- Flash memory for storing signed binaries
- Bootloaders with secure boot
- ATF secure monitor in EL3
- Linux kernel and userland
- Android virtual machines and services

The TOE requires also the following components to be properly configured and available in the operational environment:

- ATF needs to have the Trustonic SPD patches
- The platform-specific callbacks in the SPD patch must be implemented.

Table 5 specifies the minimum system requirements for the proper operation of the TOE when deployed on a hardware platform.

Category	Requirement
Processors	ARM Cortex A53 or A57, maximum 8 cores Samsung LSI Exynos7870

Category	Requirement
Flash Partition	Minimum 320 KB, recommended 512 KB
Memory	Minimum 3 MB protected DDR Optional IRAM: Minimum 116 KB, maximum 256 KB supported
HW Identity	96 bits SoC ID unique among all chips of this silicon provider
HW Key for K.DeviceFuse	256 bits SoC-specific key for the TEE
RNG	64 bits of random data on each boot for the TEE

Table 5 TOE Minimum Requirements

4.4 Usage and Major security features of the TOE

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

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

Security features provided by the Trusted Applications and the SoC are out of the scope of the TOE. The management of Trusted Applications is also out of the scope of the TOE but it is covered by objectives on the environment of the TOE.

4.4.1 Kinibi Security Features

The Kinibi security functionality in the end-user phase which is in the scope of the evaluation consists of:

4.4.1.1 Boot at power-up

Kinibi boots after the SoC has successfully powered-up and executed the ATF code. The TOE boot operations ensure the correct initialization of the TOE functionalities.

Kinibi relies on hardware roots of trust (secure boot).

Kinibi ensures the authenticity and integrity of the code running in the Secure World.

4.4.1.2 Memory Management

Kinibi manages the persistent and volatile memories of the product according to the capacities of the underlying SoC so as to control access to sensitive content protected by the TOE. TOE memory management services include memory allocation/deallocation, direct or handle-based access control, integrity checks, enciphering, pagination, etc.

Kinibi ensures that no residual information is available from memories, for instance, cleaning persistent memory upon allocation and deallocation and wiping of volatile memory upon sensitive operations.

Kinibi accepts 3 memory regions in its bootloader arguments:

1. *Secure IRAM:* The memory region is inaccessible from NWd and secure from physical attacks.
2. *Secure DRAM:* The memory region is inaccessible from NWd.
3. *Non-Secure DRAM:* The memory region is accessible from NWd.

4.4.1.3 Cryptographic operations

There are several functions within the TOE related to cryptographic support: generation of random numbers, digital signature (generation and verification), data encryption and decryption, key destruction, the generation of hash values and the generation and verification of MAC values.

4.4.1.3.1 Global Platform API

The following tables list the cryptographic algorithms that Kinibi-311A supports:

GP Algorithm name	Possible Key Sizes
TEE_TYPE_AES	128 or 256 bits
TEE_TYPE_DES	Always 56 bits
TEE_TYPE_DES3	112 bits. No support for 168 bits
TEE_TYPE_HMAC_MD5	Between 64 and 512 bits, multiple of 8 bits.
TEE_TYPE_HMAC_SHA1	Between 80 and 512 bits, multiple of 8 bits
TEE_TYPE_HMAC_SHA224	Between 112 and 512 bits, multiple of 8 bits.
TEE_TYPE_HMAC_SHA256	Between 192 and 1024 bits, multiple of 8 bits
TEE_TYPE_HMAC_SHA384	Between 256 and 1024 bits, multiple of 8 bits.
TEE_TYPE_HMAC_SHA512	Between 256 and 1024 bits, multiple of 8 bits.
TEE_TYPE_RSA_PUBLIC_KEY	Key size up to 2048 bits
TEE_TYPE_RSA_KEYPAIR	Key size up to 2048 bits
TEE_TYPE_DSA_PUBLIC_KEY	Depends on Algorithm: ALG_DSA_SHA1: Between 512 and 1024 bits, multiple of 64 bits ALG_DSA_SHA224: 2048 bits ALG_DSA_SHA256: 2048 or 3072 bits
TEE_TYPE_DSA_KEYPAIR	Same as for DSA public key size.
TEE_TYPE_DH_KEYPAIR	From 256 to 2048 bits
TEE_TYPE_ECDSA_PUBLIC_KEY TEE_TYPE_ECDSA_KEYPAIR	All the following curve sizes are supported: • TEE_ECC_CURVE_NIST_P192 • TEE_ECC_CURVE_NIST_P224 • TEE_ECC_CURVE_NIST_P256 • TEE_ECC_CURVE_NIST_P384

	• TEE_ECC_CURVE_NIST_P521
TEE_TYPE_ECDH_PUBLIC_KEY TEE_TYPE_ECDH_KEYPAIR	All the following curve sizes are supported: • TEE_ECC_CURVE_NIST_P192 • TEE_ECC_CURVE_NIST_P224 • TEE_ECC_CURVE_NIST_P256 • TEE_ECC_CURVE_NIST_P384 • TEE_ECC_CURVE_NIST_P521

Table 6: Supported GP cryptographic algorithms and key sizes

Algorithm	Possible Modes
TEE_ALG_AES_ECB_NOPAD TEE_ALG_AES_CBC_NOPAD TEE_ALG_AES_CTR	TEE_MODE_ENCRYPT TEE_MODE_DECRYPT
TEE_ALG_AES_CTS TEE_ALG_AES_XTS TEE_ALG_AES_CCM TEE_ALG_AES_GCM	Not supported
TEE_ALG_DES_ECB_NOPAD TEE_ALG_DES_CBC_NOPAD TEE_ALG_DES3_ECB_NOPAD TEE_ALG_DES3_CBC_NOPAD	TEE_MODE_ENCRYPT TEE_MODE_DECRYPT
TEE_ALG_DES_CBC_MAC_NOPAD TEE_ALG_AES_CBC_MAC_NOPAD TEE_ALG_AES_CBC_MAC_PKCS5 TEE_ALG_AES_CMAC TEE_ALG_DES_CBC_MAC_PKCS5 TEE_ALG_DES3_CBC_MAC_NOPAD TEE_ALG_DES3_CBC_MAC_PKCS5	Not supported
TEE_ALG_RSASSA_PKCS1_V1_5_MD5 TEE_ALG_RSASSA_PKCS1_V1_5_SHA1 TEE_ALG_RSASSA_PKCS1_V1_5_SHA224 TEE_ALG_RSASSA_PKCS1_V1_5_SHA256 TEE_ALG_RSASSA_PKCS1_V1_5_SHA384 TEE_ALG_RSASSA_PKCS1_V1_5_SHA512 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA1 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA224 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA256 TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA384	TEE_MODE_SIGN TEE_MODE_VERIFY

TEE_ALG_RSASSA_PKCS1_PSS_MGF1_SHA512 TEE_ALG_DSA_SHA1 TEE_ALG_DSA_SHA224 TEE_ALG_DSA_SHA256 TEE_ALG_ECDSA	
TEE_ALG_RSAES_PKCS1_V1_5 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA1 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA224 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA256 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA384 TEE_ALG_RSAES_PKCS1_OAEP_MGF1_SHA512 TEE_ALG_RSA_NOPAD	TEE_MODE_ENCRYPT TEE_MODE_DECRYPT

Table 7: Supported GP algorithmic modes

4.4.1.3.2 Legacy API

See also the [TR-DEVAPI] for more information about the abbreviations.

Algorithms in these tables have to be interpreted as a combination of cryptographic algorithm, paddings, block sizes and other information.

List of Cipher algorithms:

- AES ciphers
- Triple-DES ciphers
- DES ciphers
- RSA ciphers (maximum key size 4096 bit)

Supported Cipher Algorithm
TLAPI_ALG_AES_128_CBC_NOPAD
TLAPI_ALG_AES_128_CBC_ISO9797_M1
TLAPI_ALG_AES_128_CBC_ISO9797_M2
TLAPI_ALG_AES_128_CBC_PKCS5
TLAPI_ALG_AES_128_CBC_PKCS7
TLAPI_ALG_AES_128_ECB_NOPAD
TLAPI_ALG_AES_128_ECB_ISO9797_M1
TLAPI_ALG_AES_128_ECB_ISO9797_M2
TLAPI_ALG_AES_128_ECB_PKCS5
TLAPI_ALG_AES_128_ECB_PKCS7
TLAPI_ALG_AES_128_CTR_NOPAD
TLAPI_ALG_AES_256_CBC_NOPAD
TLAPI_ALG_AES_256_CBC_ISO9797_M1
TLAPI_ALG_AES_256_CBC_ISO9797_M2
TLAPI_ALG_AES_256_CBC_PKCS5
TLAPI_ALG_AES_256_CBC_PKCS7
TLAPI_ALG_AES_256_ECB_NOPAD
TLAPI_ALG_AES_256_ECB_ISO9797_M1
TLAPI_ALG_AES_256_ECB_ISO9797_M2
TLAPI_ALG_AES_256_ECB_PKCS5
TLAPI_ALG_AES_256_ECB_PKCS7
TLAPI_ALG_AES_256_CTR_NOPAD
TLAPI_ALG_AES_128_CBC_PKCS5
TLAPI_ALG_AES_128_CBC_PKCS7
TLAPI_ALG_AES_128_ECB_NOPAD
TLAPI_ALG_AES_128_ECB_ISO9797_M1
TLAPI_ALG_AES_128_ECB_ISO9797_M2
TLAPI_ALG_AES_128_ECB_PKCS5

Supported Cipher Algorithm
TLAPI_ALG_AES_128_ECB_PKCS7
TLAPI_ALG_AES_128_CTR_NOPAD
TLAPI_ALG_AES_256_CBC_NOPAD
TLAPI_ALG_AES_256_CBC_ISO9797_M1
TLAPI_ALG_AES_256_CBC_ISO9797_M2
TLAPI_ALG_AES_256_CBC_PKCS5
TLAPI_ALG_AES_256_CBC_PKCS7
TLAPI_ALG_AES_256_ECB_NOPAD
TLAPI_ALG_AES_256_ECB_ISO9797_M1
TLAPI_ALG_AES_256_ECB_ISO9797_M2
TLAPI_ALG_AES_256_ECB_PKCS5
TLAPI_ALG_AES_256_ECB_PKCS7
TLAPI_ALG_AES_256_CTR_NOPAD
TLAPI_ALG_AES_128_CBC_PKCS5
TLAPI_ALG_AES_128_CBC_PKCS7
TLAPI_ALG_AES_128_ECB_NOPAD
TLAPI_ALG_3DES_2KEY_CBC_ISO9797_M1
TLAPI_ALG_3DES_2KEY_CBC_ISO9797_M2
TLAPI_ALG_3DES_2KEY_CBC_NOPAD
TLAPI_ALG_3DES_2KEY_CBC_ PKCS5
TLAPI_ALG_3DES_3KEY_CBC_ISO9797_M1
TLAPI_ALG_3DES_3KEY_CBC_ISO9797_M2
TLAPI_ALG_3DES_3KEY_CBC_NOPAD
TLAPI_ALG_3DES_3KEY_CBC_ PKCS5
TLAPI_ALG_DES_CBC_ISO9797_M1
TLAPI_ALG_DES_CBC_ISO9797_M2
TLAPI_ALG_DES_CBC_NOPAD
TLAPI_ALG_DES_CBC_PKCS5

Supported Cipher Algorithm
TLAPI_ALG_DES_ECB_ISO9797_M1
TLAPI_ALG_DES_ECB_ISO9797_M2
TLAPI_ALG_DES_ECB_NOPAD
TLAPI_ALG_DES_ECB_PKCS5
TLAPI_ALG_RSA_ISO14888
TLAPI_ALG_RSA_NOPAD
TLAPI_ALG_RSA_PKCS1
TLAPI_ALG_RSA_SHA1_OAEP
TLAPI_ALG_RSA_SHA224_OAEP
TLAPI_ALG_RSA_SHA256_OAEP
TLAPI_ALG_RSA_SHA384_OAEP
TLAPI_ALG_RSA_SHA512_OAEP
TLAPI_ALG_RSACRT_SHA1_OAEP
TLAPI_ALG_RSACRT_SHA224_OAEP
TLAPI_ALG_RSACRT_SHA256_OAEP
TLAPI_ALG_RSACRT_SHA384_OAEP
TLAPI_ALG_RSACRT_SHA512_OAEP

Table 8: Supported Cipher Algorithms in Legacy API

List of Message Digest algorithms:

Supported Digest Algorithms
TLAPI_ALG_MD5
TLAPI_ALG_SHA1
TLAPI_ALG_SHA256
TLAPI_ALG_SHA224
TLAPI_ALG_SHA384
TLAPI_ALG_SHA512

Table 9 Supported Digest Algorithms in Legacy API

The DSA key sizes from 512 to 3072 bits are supported.

Supported elliptic curve types:

- ECC_CURVE_NIST_P192,
- ECC_CURVE_NIST_P224,
- ECC_CURVE_NIST_P256,
- ECC_CURVE_NIST_P384
- ECC_CURVE_NIST_P521.

List of Signature algorithms:

Enumerator Name
TLAPI_ALG_DES_MAC4_NOPAD
TLAPI_ALG_DES_MAC4_PKCS5
TLAPI_ALG_DES_MAC8_ISO9797_1_M2_ALG3
TLAPI_ALG_DES_MAC8_ISO9797_M1
TLAPI_ALG_DES_MAC8_ISO9797_M2
TLAPI_ALG_DES_MAC8_NOPAD
TLAPI_ALG_DES_MAC8_PKCS5
TLAPI_ALG_HMAC_SHA_256
TLAPI_ALG_HMAC_SHA1
TLAPI_ALG_HMAC_SHA224
TLAPI_ALG_HMAC_SHA256
TLAPI_ALG_HMAC_SHA384
TLAPI_ALG_HMAC_SHA512
TLAPI_ALG_HMAC_MD5
TLAPI_SIG_RSA_SHA_ISO9796
TLAPI_SIG_RSA_SHA_ISO9796_MR
TLAPI_SIG_RSA_SHA_PKCS1
TLAPI_SIG_RSA_SHA256_PSS
TLAPI_SIG_RSA_SHA1_PSS
TLAPI_SIG_RSA_SHA1_PKCS1

TLAPI_SIG_RSA_SHA224_PKCS1
TLAPI_SIG_RSA_SHA256_PKCS1
TLAPI_SIG_RSA_SHA384_PKCS1
TLAPI_SIG_RSA_SHA512_PKCS1
TLAPI_SIG_RSA_SHA224_PSS
TLAPI_SIG_RSA_SHA384_PSS

TLAPI_SIG_RSA_SHA512_PSS
TLAPI_SIG_DSA_RAW
TLAPI_SIG_DSA_HASHED
TLAPI_SIG_ECDSA_RAW
TLAPI_SIG_ECDSA_HASHED

Table 10 Supported Signature Algorithms in Legacy API

4.4.1.4 Key Management

Kinibi provides secure generation, destruction, replacement and storage of cryptographic keys.

4.4.1.5 RNG

Kinibi provides a random number generator with sufficient entropy for cryptographic purposes. The Random Number Generation is conformant to NIST SP 800-90A.

4.4.1.6 Atomic Operations

Kinibi performs atomic operations, for instance, writing or erasing of individual or multiple storage locations. The TOE guarantees that atomic operations are either performed completely or have no effect in case of interruption.

4.4.1.7 Isolation mechanism

Kinibi ensures:

- Isolation of the TEE services, the TEE resources involved and all the TAs and TDrivers from the REE.
- Isolation between TAs and isolation of the TEE from TAs.

Kinibi implements several mechanisms to achieve isolation:

1. CPU register saving and restoring on context switches.
2. Virtual memory to ensure TAs and TDrivers cannot access arbitrary memory.
3. Restricted memory management via the TA API.
4. Driver memory management APIs do not allow to mark memory executable.

Thanks to these isolation mecanisms, Kinibi guarantees the correct execution of TDriver and TA code.

4.4.1.8 Code verification

Kinibi verifies the authenticity of all code of additional drivers and of Trusted Applications before execution.

4.4.1.9 Services to the Trusted Application

Kinibi provides security services to Trusted Application through the GlobalPlatform TEE Internal API [GPIAPI] and Trustonic's proprietary APIs defined in [TR-DEVAPI].

4.4.1.10 Confidentiality and Integrity of Data

Kinibi offers a mechanism to wrap data into Secure Objects, which contain the data in encrypted and integrity protected form. Secure Objects are passed by the TAs to the Rich OS client application and can be stored in an arbitrary insecure location.

When the data is needed again, the Secure Object must be returned to the TA for the unwrapping of data.

Every Secure Object is encrypted with a key derived from the device master key K.Device.Fuse and other specific context data. For every TA, a unique key is derived, and within the TA, different keys can be derived, too. The keys are managed by Kinibi and cannot be exported. This ensures that any wrapped data cannot be un-wrapped by any other party besides the platform that has generated it.

4.4.1.11 Trusted Storage

Trusted storage service ensures:

- The confidentiality of the stored data and keys
- The authenticity of the stored data and keys
- The consistency of each TA stored data and keys
- The atomicity of the operations that modify the storage
- Device binding

Kinibi ensures trusted storage of TA and TEE data and keys.

4.4.1.12 User Identification

Kinibi implements proprietary functionalities and user identity verification to ensure that the identity of a TA is guaranteed from others TAs and CAs. "User" stands for CA, TA or TEE.

Kinibi defines several types of Trusted Applications:

- Global-Platform-type TAs
- Proprietary TAs

Kinibi defines several types of Secure Services:

- Service Provider TA (SPTA)
- System TA (SysTA)
- Middleware (MW)
- Trusted Driver (TD)

Drivers, Middleware and System TAs are developed by Trustonic and by the SIP and the OEMs to integrate Kinibi with the REE. They are pre-installed in devices (but not necessarily in the TEE) during production. SPTAs are developed by service providers and can be installed on devices after production.

The identity of a Client Application is ensured by the REE. The TEE reports to the TA the identity of the CA as reported by the REE. However, the quality of the CA identity depends on the REE that by definition is less trusted than the TEE.

4.4.1.13 Security Management

The management of the security critical parameters of the TOE is performed by authorized users. Security parameters are stored in specific files that are protected by the access control mechanisms of the TOE against unauthorized access by users that are not administrative users.

4.4.1.14 Device identification

The TOE provides a service for storing and retrieving the unique device identifier.

4.4.1.15 Secure time

The TOE provides a TEE time which is monotonic between resets, and in particular monotonic during the lifetime of a TA instance.

4.4.1.16 TA management

The TOE can load Trusted Applications at runtime from a signed image provided by the REE. Such a load operation constitutes a TA installation operation, and causes an instance of the TA to be spawned. When the TA instance dies, the TA is unloaded from memory; this constitutes a TA uninstallation operation.

The infrastructure that produces signed TA images is outside the scope of the TOE. It comprises both on-device and off-device components which are presented in [TT-KSA].

4.4.2 Kinibi Intended 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] and Trustonic White Paper: [WP-JEE14] for an overview of the main TEE use cases.

4.5 TOE Life Cycle

The TOE life cycle distinguishes stages for development, SIP integration, OEM integration, device production (device assembling) and operational use (end-usage of the device). The development and production of the TOE (cf. CC part 1 [CC1], para.139) together constitute the development phase of the TOE. The development phase is subject of CC evaluation according to the assurance life cycle (ALC) class. The TOE life cycle is split in six phases:

- Phase 0 corresponds to the development of the environment of the TOE: design of hardware, firmware and software of the TEE and the REE; it does not cover Kinibi, however a dummy TEE OS is a result of this phase.
- Phase 1 corresponds to the Kinibi design and development.
- Phase 2 corresponds to the Kinibi porting for a specific Silicon Provider SoC.
- Phase 3 corresponds to the SIP and OEM integration, validation and preparation of the software to load in the product that will include the TEE, any pre-installed Trusted Application and Trusted Drivers, and additional software required to use the product (e.g. REE, Client Applications).
- Phase 4 covers Kinibi flashing on the hardware and RoT injection, 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 5 stands for the end-usage of the device

The development of the environment of the TOE is performed during Phase 0. This includes first the Hardware/Firmware conception, design, implementation, testing and documentation; then the REE software conception, design, implementation, testing and documentation; The SIP produces a development board that contains a release candidate version of the SoC, the REE and a dummy or legacy version of the TOE.

The development of the Kinibi generic product is performed during Phase 1. This phase includes Kinibi design, testing and documentation. The development is made in a controlled environment that avoids disclosure of source code, data and any critical documentation and that guarantees the integrity of these elements.

The porting of Kinibi to a specific SoC occurs in phase 2. The SIP and Trustonic usually have a workshop in a controlled environment and integrate the Kinibi OS on the development board. The SIP delivers the development board, the SIP Board Support Package (BSP) including the REE software and bootloaders as well as some documentation to Trustonic. Trustonic then adopts the platform-specific parts of Kinibi to this chip and integrates the REE components into

the REE. Trustonic then ensures that the board boots with Kinibi and that all the validation tests pass.

The delivery of Kinibi occurs at the end of phase 2. In this phase, Kinibi is delivered by Trustonic to the SIP development center. Note that the SIP does not itself modify the Kinibi image any further.

Delivery and acceptance procedures guaranty the authenticity, the confidentiality and integrity of the exchanged pieces. Kinibi delivery involves encrypted signed sending and it supposes the previous exchange of public keys. The evaluation of the TOE includes the delivery process.

The OEM and the SIP develop, prepare and validate the software of the final product, including the TOE (the Kinibi OS), additional Trusted Drivers, preinstalled TAs and the REE in Phase 3.

Note that many secure services are already developed and integrated into the final device without involvement of additional third-party service providers. Today, common REEs require a TEE and it is a joint work of the OEM, the SIP and Trustonic to develop respective TAs on top of Kinibi that fulfill the REE requirements. In addition, the OEMs add secure services to Kinibi to differentiate from competitors. Also the SIPs usually add secure services that are integrated more closely with their chips, and use for example the crypto accelerator hardware.

Most secure services require an architecture that includes Trusted Applications and Trusted Drivers. When developing TAs, the respective developers can follow the guidance AGD_OPE. When developing Trusted Drivers, developers have to follow the guidance AGD_PRE to maintain the security objectives for the environment.

The OEM prepares and installs the Kinibi image on the flash of the final device in phase 4, though it may be upgraded later. The OEM puts pre-installed secure services in the filesystem of the REE. During early production phases the device is not booted and only partitions with the REE, TEE and TA software are flashed to the device. In later production phases the device is booted and the root of trust of the TEE storage services is set (injection or on-board creation) in phase 4.

Phase 4 takes place in the OEM Manufacturing site that is a controlled environment (secure locations, secure procedures and trusted personnel). The product is tested and all critical materials including data, test suites and documentation are protected from disclosure and modification.

In phase 5, Kinibi provides the full set of security functionalities of the TEE. At the end of this phase, Kinibi is ready for use, and the device is sent to the customer.

Trusted Applications Management, which may include loading, installation, deletion and also personalization generally occur in phase 5.

The TOE delivery point establishes the limits of the evaluation:

- The security of the environments, processes and procedures before the delivery point is evaluated according to the ALC Class
- The security of the environments processes and procedures from the delivery point up to end of phase 5 are out of the scope of the evaluation. They are usually covered by organizational security policies and security objectives for the environment
- The security of the end-usage environment is covered by the TOE security functionalities, by security objectives for the environment and the guidance AGD_OPE.

Table 11 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.

Phases	**Actors**
(Informative — out of scope for ALC) Phase 0: Firmware/Hardware design, REE development	The TEE hardware designer is in charge of designing the processor(s) where the TEE software runs and designing (part of) the hardware security resources used by the TEE. The silicon vendor designs the ROM code and the secure portion of the TEE chipset. This phase is out of scope for this security target since it only concerns the preparation of the hardware which not part of the TOE.
Phase 1: TEE Software design & Development	The TEE software developer • Is in charge of TEE software development and testing • also develop the TEE initialization code that instantiates/initializes the TEE • Develops the REE integration software
Phase 2: TEE software porting & delivery	The TEE SIP integration developer • Is in charge of adopting the TEE to a SoC • Secure delivery of the TEE to the SIP
Phase 3: SIP and OEM integration	The silicon vendor produces the TEE chipset. 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).
Phase 4: Device manufacturing + TEE installation and RoT injection	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.
Phase 5: 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 11 Actors in the Device Life Cycle

5 SECURITY PROBLEM DEFINITION

This chapter introduces the security problem addressed by the TOE 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.

5.1 Assets

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

TEE identification

TEE identification data that is globally unique among all GlobalPlatform TEEs whatever the manufacturer, vendor or integrator.

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.

TA code

The code of the installed Trusted Applications.

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.

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.

Properties: integrity and confidentiality, including random numbers generated by the TEE.

TEE persistent data

TEE persistent data, including TEE keys and TA properties.

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

TEE storage root of trust

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

Properties: uniqueness, integrity and confidentiality.

RNG

Random Number Generator based on Random data output provided by the SoC.

Properties: unpredictable random numbers, sufficient entropy.

TEE software initialization data

Initialization data used for starting up the TEE in a secure state and to the complete activation of the TEE security services.

Properties: integrity.

5.2 Users / Subjects

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

Trusted Application (TA)

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

Rich Execution Environment (REE)

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

5.3 Threats

This Security Target addresses 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 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 Security Target 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 at 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).

No assumption holds on the identification phase regarding the means of the attacker, software or hardware, and the possibility to use more than one device, potentially in a destructive way. Henceforth, the profile of the attacker at the identification phase is given by the overall attack potential that has been selected for the TEE and in particular the Trusted OS: Enhanced-Basic (the upper mark is reached when the exploitation part of the attack does not give any point). 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, the choice of Enhanced-Basic attack potential means that advanced attackers able to act at deep package and SoC levels are excluded.

On the other hand, two main attacker profiles may arise at 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.

The overall attack potential strongly limits the possibility to face more advanced attackers at exploitation phase, provided that, in general, the identification phase requires at least the same capabilities for each applicable factor (e.g. expertise, resources and spent time). We refer to the Annex A of [PP-TEE] 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.

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

T.ABUSE_FUNCT

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

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

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

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

Application Note:

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

In particular a fake application running in the Rich OS which masquerades as a security application running in the TEE can grab PINs and passwords and run the real security application on behalf of the user. However, such threat is not countered by the TEE alone and must be taken into account in the design of the use case, for instance by using an applicative authenticated communication channel between the client and the TA.

T.CLONE

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

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

T.FLASH_DUMP

An attacker partially or totally recovers the content of the external Flash in 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 process.

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.

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.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.

5.4 Organizational Security Policies

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

OSP.TEE_ID

Generation of the TEE identifier, outside or inside the TEE, shall enforce the statistical uniqueness of this value. The TEE shall provide means to store and retrieve this identifier.

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.

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).

OSP.TA_MANAGEMENT

TA management software shall be developped by competent and trustworthy individuals, who shall ensure that the TA management software respects the security guarantees stated in [PRE], that the additional software does not break any security guarantee of the TOE, and that it complies with the TA development guidelines.

5.5 Assumptions

This section states the assumptions that hold on the TEE operational environment. These assumptions have to be met by the operational environment but are out of the scope of the evaluation.

A.Configuration

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

A.CONNECT

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

All connections to and from the Hardware/Firmware or the software environment not protected by the TSF itself are physically or logically protected within the TOE environment to ensure the integrity and confidentiality of the data transmitted.

A.PEER.FUNC

Hardware/Firmware and the software environment trusted by the TSF to provide TSF data or services to the TOE, or to support the TSF in the enforcement of security policy decisions are assumed to correctly implement the functionality used by the TSF consistent with the assumptions defined for this functionality.

A.PEER.MGT

Hardware/Firmware and the software environment trusted by the TSF to provide TSF data or services to the TOE, or to support the TSF in the enforcement of security policy decisions are assumed to be under the same management control and operate under security policy constraints compatible with those of the TOE.

A.Power-Up

It is assumed that the TOE is started through a secure initialization process (starting with on-chip ROM code) that ensures the integrity of the Trusted OS firmware and the root of trust of the TEE storage and is bound to the SoC of the device.

A.RNG

It is assumed that the the SoC provides a random number generator suitable as an entropy source as specified in NIST SP 800-90A. Random numbers output by this generator is not predictable and have sufficient entropy.

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.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 [AGD]. For instance, for installation, pre-personalization or personalization guides, only the described set-up configurations or personalization profiles are covered by the certificate.

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.

6 SECURITY OBJECTIVES

6.1 Security Objectives for the TOE

This section states the security objectives for the TOE. The objectives for the TOE concern Kinibi and its applicative behaviour if any only, according with the definition of the TOE given in section 4.1.

O.CA_TA_IDENTIFICATION

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

Application Note:

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

- The Client identity of 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.TEE_ID

The TOE shall provide means to retrieve the unique TEE identifier and shall ensure that it is non-modifiable.

O.INSTANCE_TIME

The TOE 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 TOE 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. [GPCAPI], [TR-DEVAPI]). 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. [TR-DEVAPI], [GPIAPI] 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 TOE shall ensure the cryptographic quality of random number generation. Random numbers shall not be predictable and shall have sufficient entropy.

O.RUNTIME_CONFIDENTIALITY

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

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

O.TA_ISOLATION

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

Application Note:

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

O.TEE_DATA_PROTECTION

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

O.TEE_ISOLATION

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

Application Note:

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

O.TRUSTED_STORAGE

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

- The confidentiality of the stored data and keys is enforced
- The authenticity of the stored data and keys is enforced
- The consistency of the 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.

O.TA_AUTHENTICITY

The TOE shall verify code authenticity of Trusted Applications.

6.2 Security Objectives for the Operational Environment

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

The security objectives for the TOE operational environment concern the product operational environment, including the SoC, the ATF and other other hardware, firmware and software trusted by the TSF. The properties of trusted hardware, firmware and software are stated as security objectives for the environment, without SFR counterpart.

OE.INTEGRATION_CONFIGURATION

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

OE.PROTECTION_AFTER_DELIVERY

The TOE shall be protected by the environment after delivery and before entering the final usage phase. The persons manipulating the TOE in the operational environment shall apply the TOE guidance [AGD]. 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.

OE.UNIQUE_TEE_ID

Generation of the TEE identifier, outside or inside the TEE, shall enforce the statistical uniqueness of this data.

OE.Configuration

The TOE shall be installed and configured properly for starting up the TOE in a secure state. The Hardware/Firmware and the software environment components shall be installed and configured in a secure manner supporting the security mechanisms provided by the TOE.

OE.TRUSTED_HARDWARE

Hardware/Firmware implement the protocols and mechanisms required by the TSF to support the enforcement of the security policy. Those systems provide the functions required by the TOE (e.g. the device individual symmetric master key) and are sufficiently protected from any attack that may cause those functions to provide false results.

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

OE.TRUSTED_FIRMWARE

Developers of trusted firmware and drivers are competent and trustworthy. They are capable and willing to ensure that the additional software does not break any security guarantee of the TOE, and they actually to so. Driver developers comply with the driver development guidelines set by the Trusted OS provider, in addition to the TA development guidelines which also hold for drivers.

OE.TA_MANAGEMENT

Developers of TA management software are competent and trustworthy. They are capable and willing to ensure that the TA management software ensures that only trusted entities can deploy privileged Trusted Applications and that only the owner of a TA identity can deploy a TA bearing this identity, and they actually do so. They are capable and willing to

ensure that the additional software does not break any security guarantee of the TOE, and they actually to so. Developers of TA management software comply with the TA development guidelines.

OE.RNG

The SoC shall provide a random number generator suitable as an entropy source as specified in NIST SP 800-90A. Random numbers output by this generator shall not be predictable and shall have sufficient entropy. The SOC shall ensure that no information about the produced random numbers is available to an attacker since they might be used to generate cryptographic keys.

OE.INITIALIZATION

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

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

6.3 Security Objectives Rationale

6.3.1 Threats

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

- OE. INITIALIZATION ensures that the TEE security functionality is correctly initialized
- O.OPERATION ensures correct operation of the security functionality and of the underlying state machines 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)
- OE.DEBUG ensures that TOE debug features cannot be used as vectors for abusing TOE functionality
- 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 device identification means
- OE. INITIALIZATION ensures that the TOE 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 of security functionality at runtime of security functionalities or data used to detect or prevent cloning
- O.TEE_DATA_PROTECTION prevents the TEE from using TEE data that is inconsistent or not authentic
- O.TRUSTED_STORAGE ensures that the trusted storage is bound to the device and prevents the TEE from using data that is inconsistent or not authentic
- OE.UNIQUE_TEE_ID ensures that the device ID is in fact unique.

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.
- OE.TA_MANAGEMENT ensures the authenticity and integrity of the TAs installed in the TEE

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 prior to execution
- 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.TA_MANAGEMENT ensures the authenticity and integrity of the TAs installed in the TEE
- OE.PROTECTION_AFTER_DELIVERY covers the import of foreign code in a phase other than the end-user phase.

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

- OE.INITIALIZATION ensures that the TOE 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.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime
- O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified prior to execution
- O.TA_ISOLATION ensures the separation of the TA
- O.TEE_DATA_PROTECTION covers persistent TEE data which might influence the behavior of the TEE
- O.TEE_ISOLATION enforces the separation between the TEE and the outside (REE and TAs).

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

- OE.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
- OE.TRUSTED_HARDWARE ensures that the memory barriers set up by the TOE work as desired
- OE.TRUSTED_FIRMWARE ensures that the trusted part of the TOE environment does not bypass TOE security functionality.

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

- OE.INITIALIZATION ensures the correct initialization of the TOE 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
- OE.RNG provides an entropy source for 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.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.

6.3.2 Organizational Security Policies

OSP.TEE_ID The objective O.TEE_ID directly covers this OSP. If the TEE identifier is generated outside the TEE, the objective OE.UNIQUE_TEE_ID ensures that the TEE identifier is unique. If the TEE identifier is generated inside the TEE, the objective O.RNG ensures the statistical uniqueness of the identifier.

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

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

OSP.TA_MANAGEMENT The objective OE.CRYPTOGRAPHY directly covers this OSP.

6.3.3 Assumptions

A.Configuration The assumption on the IT environment to properly install and configure the TOE is covered by OE.Configuration requiring properly installation and configuration for starting up the TOE in a secure state.

A.CONNECT The assumption that all connections to and from Hardware/Firmware or the software environment not protected by the TSF itself are physically or logically protected is covered by OE.TRUSTED_HARDWARE:

- requiring that Hardware/Firmware or the software environment provide the functions required by the TOE and are sufficiently protected from any attack that may cause those functions to provide false results.
- demanding the physical and logical protection equivalent to the TOE.

A.PEER.FUNC The assumption on Hardware/Firmware to correctly implement the functionality used by the TSF consistent with the assumptions defined for this functionality is covered by OE.TRUSTED_HARDWARE requiring that the Hardware/Firmware implement the protocols

and mechanisms required by the TSF to support the enforcement of the security policy, and OE.TRUSTED_FIRMWARE requiring that additional Firmware/Software (such as drivers) does not interfere with the security policy enforced by the TSF.

A.PEER.MGT The assumption on Hardware/Firmware to be under the same management control and operate under security policy constraints compatible with those of the TOE is covered by OE.TRUSTED_HARDWARE requiring that the Hardware/Firmware are under the same management domain as the TOE, and are managed based on the same rules and policies applicable to the TOE.

A.Power-Up OE.INITIALIZATION satisfies the assumption that the TOE is started through a secure initialization process.

A.RNG OE.RNG satisfies the assumption that the environment provides a sufficient entropy and non-predictable RNG to the TOE.

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

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

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

6.3.4 SPD and Security Objectives

Threats	Security Objectives	Rationale
T.ABUSE_FUNCT	OE.INITIALIZATION, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TEE_ISOLATION, OE.TA_DEVELOPMENT	Section 2.3.1
T.CLONE	O.TEE_ID, OE. INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, OE.UNIQUE_TEE_ID	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, OE.TA_MANAGEMENT	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	Section 2.3.1

Threats	Security Objectives	Rationale
T.PERTURBATION	OE.INITIALIZATION, O.INSTANCE_TIME, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_DATA_PROTECTION, O.TEE_ISOLATION, O.TA_AUTHENTICITY, OE.INITIALIZATION	Section 2.3.1
T.RAM	OE.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION	Section 2.3.1
T.RNG	OE.INITIALIZATION, O.RNG, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY	Section 2.3.1
T.SPY	O.TA_ISOLATION, O.TEE_ISOLATION, O.TRUSTED_STORAGE, O.RUNTIME_CONFIDENTIALITY	Section 2.3.1
T.STORAGE_CORRUPTION	O.OPERATION, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, OE.ROLLBACK	Section 2.3.1

Table 12 Threats and Security Objectives - Coverage

Security Objectives	Threats	Rationale
O.CA_TA_IDENTIFICATION	T.IMPERSONATION	
O.TEE_ID	T.CLONE	
O.INSTANCE_TIME	T.PERTURBATION	
O.OPERATION	T.ABUSE_FUNCT, T.IMPERSONATION, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.STORAGE_CORRUPTION	
O.RNG	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_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	

Security Objectives	Threats	Rationale
O.TRUSTED_STORAGE	T.CLONE, T.FLASH_DUMP, T.ROGUE_CODE_EXECUTION, T.SPY, T.STORAGE_CORRUPTION	
OE.INITIALIZATION	T.ABUSE_FUNCT, T.CLONE, T.PERTURBATION, T.RAM, T.RNG	
O.TA_AUTHENTICITY	T.ROGUE_CODE_EXECUTION, T.PERTURBATION	
OE.INTEGRATION_CONFIGURATION	T.ROGUE_CODE_EXECUTION	
OE.PROTECTION_AFTER_DELIVERY	T.ROGUE_CODE_EXECUTION	
OE.ROLLBACK	T.STORAGE_CORRUPTION	
OE.SECRETS		
OE.TA_DEVELOPMENT	T.ABUSE_FUNCT	
OE.UNIQUE_TEE_ID	T.CLONE	
OE.Configuration		
OE.TRUSTED_HARDWARE		
OE.TRUSTED_FIRMWARE		
OE.TA_MANAGEMENT		
OE.RNG		
OE.INITIALIZATION		

Table 13 Security Objectives and Threats - Coverage

Organisational Security Policies	Security Objectives	Rationale
OSP.TEE_ID	O.TEE_ID, OE.UNIQUE_TEE_ID, O.RNG	Section 2.3.2
OSP.INTEGRATION_CONFIGURATION	OE.INTEGRATION_CONFIGURATION	Section 2.3.2
OSP.SECRETS	OE.SECRETS	Section 2.3.2
OSP.TA_MANAGEMENT	OE.TA_MANAGEMENT	Section 2.3.2

Table 14 OSPs and Security Objectives - Coverage

Security Objectives	Organisational Security Policies	Rationale
O.CA_TA_IDENTIFICATION		
O.TEE_ID	OSP.TEE_ID	
O.INSTANCE_TIME		
O.OPERATION		
O.RNG	OSP.TEE_ID	
O.RUNTIME_CONFIDENTIALITY		
O.RUNTIME_INTEGRITY		

Security Objectives	Organisational Security Policies	Rationale
O.TA_ISOLATION		
O.TEE_DATA_PROTECTION		
O.TEE_ISOLATION		
O.TRUSTED_STORAGE		
O.TA_AUTHENTICITY		
OE.INTEGRATION_CONFIGURATION	OSP.INTEGRATION_CONFIGURATION	
OE.PROTECTION_AFTER_DELIVERY		
OE.ROLLBACK		
OE.SECRETS	OSP.SECRETS	
OE.TA_DEVELOPMENT		
OE.UNIQUE_TEE_ID	OSP.TEE_ID	
OE.Configuration		
OE.TRUSTED_HARDWARE		
OE.TRUSTED_FIRMWARE		
OE.TA_MANAGEMENT	OSP.TA_MANAGEMENT	
OE.RNG		
OE.INITIALIZATION		

Table 15 Security Objectives and OSPs - Coverage

Assumptions	Security Objectives for the Operational Environment	Rationale
A.Configuration	OE.Configuration	Section 2.3.3
A.CONNECT	OE.TRUSTED_HARDWARE	Section 2.3.3
A.PEER.FUNC	OE.TRUSTED_HARDWARE	Section 2.3.3
A.PEER.MGT	OE.TRUSTED_HARDWARE	Section 2.3.3
A.Power-Up	OE.INITIALIZATION	Section 2.3.3
A.RNG	OE.RNG	Section 2.3.3
A.ROLLBACK	OE.ROLLBACK	Section 2.3.3
A.PROTECTION_AFTER_DELIVERY	OE.PROTECTION_AFTER_DELIVERY	Section 2.3.3
A.TA_DEVELOPMENT	OE.TA_DEVELOPMENT	Section 2.3.3

Table 16 Assumptions and Security Objectives for the Operational Environment - Coverage

Security Objectives for the Operational Environment	Assumptions	Rationale

Security Objectives for the Operational Environment	Assumptions	Rationale
OE.INTEGRATION_CONFIGURATION		
OE.PROTECTION_AFTER_DELIVERY	A.PROTECTION_AFTER_DELIVERY	
OE.ROLLBACK	A.ROLLBACK	
OE.SECRETS		
OE.TA_DEVELOPMENT	A.TA_DEVELOPMENT	
OE.UNIQUE_TEE_ID		
OE.Configuration	A.Configuration	
OE.TRUSTED_HARDWARE	A.CONNECT, A.PEER.FUNC, A.PEER.MGT	
OE.TRUSTED_FIRMWARE	A.PEER.FUNC	
OE.TA_MANAGEMENT		
OE.RNG	A.RNG	
OE.INITIALIZATION	A.Power-Up	

Table 17 Security Objectives for the Operational Environment and Assumptions - Coverage

7 EXTENDED REQUIREMENTS

7.1 Extended Families

7.1.1 Extended Family FCS_RNG - Generation of random numbers

7.1.1.1 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.

7.1.1.2 Extended Components

7.1.1.2.1 Extended Component FCS_RNG.1

1.1.1.1.1.1 Description

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

Management

No management activities are foreseen.

Audit

No actions are defined to be auditable.

1.1.1.1.1.2 Definition

FCS_RNG.1 Random numbers generation

FCS_RNG.1.1 The TSF shall provide a [selection: physical, non-physical true, deterministic, hybrid] 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.

7.1.2 Extended Family FPT_INI - TSF initialisation

7.1.2.1 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.

7.1.2.2 Extended Components

7.1.2.2.1 Extended Component FPT_INI.1

1.1.1.1.1.3 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.

1.1.1.1.1.4 Definition

FPT_INI.1 TSF initialisation

FPT_INI.1.1 The TOE initialization function shall verify **[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.

7.1.3 Extended Family AVA_TEE - Vulnerability analysis of TEE

7.1.3.1 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 of [PP-TEE]. 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.

Component Levelling: The family AVA_TEE only contains one component, AVA_TEE.2

7.1.3.2 Extended Components

7.1.3.2.1 Extended Component AVA_TEE.2

1.1.1.1.1.5 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.

1.1.1.1.1.6 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) and (ADV_FSP.2) and (ADV_TDS.1) and (AGD_OPE.1) and (AGD_PRE.1)

8 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.

8.1 Security Functional Requirements

All of the following SFRs are derived from the [PP-TEE].Although this ST does not claim any compliance to the [PP-TEE](as not all SFRs of the [PP-TEE] are included), the SFRs taken from [PP-TEE] are instantiated exactly as the [PP-TEE] would require it.

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

The SFRs concern Kinibi and its applicative behaviour if any only, according with the definition of the TOE given in section 4.

8.1.1 Definitions

This Security Target 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_CKM.1 Cryptographic key generation
- FCS_CKM.4 Cryptographic key destruction
- 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_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
- FMT_MTD.1 Management of TSF data
- FPT_TEE.1 Testing of external entities

The following extended security functional components defined in Chapter 7 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"
- Trusted Firmware.

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"
- 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.
- 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)

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

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

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

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

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

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

Other operations:

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

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

Runtime Data Information Flow Control SFP:

- Purpose: To control the flow of runtime data from and to executable entities and memory. This policy contributes to ensure the integrity and confidentiality of runtime data
- Subjects: S.TA_INSTANCE, S.TA_INSTANCE_SESSION, S.API, S.COMM_AGENT, S.RESOURCE, S.RAM_UNIT
- Information: I.RUNTIME_DATA
- Security attributes: S.RESOURCE.state, S.RAM_UNIT.rights, S.API.caller and I.RUNTIME_DATA.owner
- SFR instances: FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FMT_MSA.1/Runtime, FMT_MSA.3/Runtime, 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
- 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.

8.1.2 Identification

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, Trusted Application or Trusted Firmware.

FIA_USB.1 User-subject binding

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

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

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

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

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

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

Application Note:

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

FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles

- **TSF or Trusted Firmware**
- **TA_User**.

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

Application Note:

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

FIA_ATD.1 User attribute definition

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

Application Note:

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

- CA_identity: The availability of this attribute is that 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.

FAU_SAR.1 Audit review

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

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

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 TEE identifier is computed in the TOE from hardware-protected data: one-time programmable memory, retrieved via the trusted firmware.

8.1.3 Confidentiality, Integrity and Isolation

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 to** the following objects: **TEE and TA memory** and upon the **deallocation of the resource from** the following objects: **TA cryptographic and storage objects**.

FMT_MSA.1/Runtime Management of security attributes

FMT_MSA.1.1/Runtime The TSF shall enforce the **Runtime Data Information Flow Control** to restrict the ability to **change_default, query and modify** the security attributes **S.RESOURCE.state, S.RAM_UNIT.rights, S.API.caller and I.RUNTIME_DATA.owner** to

- **query I.RUNTIME_DATA.owner to the TSF role**
- **change_default, query and modify S.RESOURCE.state, S.RAM_UNIT.rights, S.API.caller to TA_User role**.

FMT_MSA.3/Runtime Static attribute initialisation

FMT_MSA.3.1/Runtime The TSF shall enforce the **Runtime Data Information Flow Control** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/Runtime The TSF shall allow the **none** to specify alternative initial values to override the default values when an object or information is created.

FDP_IFC.2/Runtime Complete information flow control

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

FDP_IFF.1/Runtime Simple security attributes

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

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

- **Rules for information flow between S.TA_INSTANCE and S.RAM_UNIT:**
  - **Flow of I.RUNTIME_DATA from S.TA_INSTANCE to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(S.TA_INSTANCE) is Write or ReadWrite**
  - **Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.TA_INSTANCE is allowed only if S.RAM_UNIT.rights(S.TA_INSTANCE) is Read or ReadWrite**
- **Rules for information flow from and to S.COMM_AGENT:**
  - **Flow of I.RUNTIME_DATA from S.COMM_AGENT to S.RAM_UNIT is allowed only if S.RAM_UNIT.rights(REE) is Write or ReadWrite**
  - **Flow of I.RUNTIME_DATA from S.RAM_UNIT to S.COMM_AGENT is allowed only if S.RAM_UNIT.rights(REE) is Read or ReadWrite**

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

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

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

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

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

FPT_ITT.1/Runtime Basic internal TSF data transfer protection

FPT_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 **separate** parts of the TOE.

Application Note:

This SFR is shared between SW and HW: relies on correct TZ firewall configuration and operation

8.1.4 Cryptography

FCS_CKM.1 Cryptographic key generation

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm and specified cryptographic key sizes that meet the following: **algorithms and key sizes listed in [CAK]**.

FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **physical deletion of key value by overwriting with a constant pattern** that meets the following: **none**.

FCS_COP.1 Cryptographic operation

FCS_COP.1.1 The TSF shall perform **cryptographic operations** in accordance with a specified cryptographic algorithm and cryptographic key sizes that meets the following: **algorithms, key sizes and operations listed in [CAK]**.

8.1.5 Initialization, Operation and Firmware Integrity

FDP_SDI.2 Stored data integrity monitoring and action

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

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **behave in a manner that does not depend on the compromised data**.

Refinement:

- Upon detection of TEE data authenticity or consistency errors, the TSF shall **behave in a manner 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**
  - **Behave in a manner that does not depend on the compromised data**.

FPT_INI.1 TSF initialisation

FPT_INI.1.1 The TOE initialization function shall verify

- **the integrity of the storage root of trust**

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:

The bootloader chain is not managed directly by Kinibi. At the stage when the TOE is loaded into RAM, it is assumed that all cryptographic signatures have been verified and that these verifications are protected enough (OE.INITIALIZATION, OE.Configuration). For this ST, the initialization phase begins when the Kinibi bootloader is started and lasts until Kinibi is in its operational and secure state.

FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 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 none** to check the fulfillment of **authenticity of TA code**.

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

FAU_ARP.1 Security alarms

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

Refinement:

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

- detection of consistency or authenticity violation of TA data, TA code or TEE data: the invalid data or code is rejected. The secure storage content is only authenticated when reading requested data. If corrupted data is requested, the request fails and an error code is returned to the requestor. There is no recovery mechanism.

- detection of TEE firmware integrity violation: Kinibi will enter a halt state if it detects an impossible system state. In the halt state, Kinibi does not execute any TA code.

Application Note:

The responsibility of TEE firmware integrity violation detection during initialization falls on the Trusted Firmware, which is outside the TOE.

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

Application Note:

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

8.1.6 Trusted Storage

FDP_ACC.1/Trusted Storage Subset access control

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

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

FDP_ACF.1/Trusted Storage Security attribute based access control

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

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

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

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

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

- **Any access to a trusted storage attempted from S.API without valid caller (S.API.caller = undefined)**
- **Any access to a trusted storage that was bound to a different TEE (with different OB.SRT)**
- **Any access to a trusted storage from a subject different from S.API**
- **Any access to a trusted storage which fails to be authenticated or which is not consistent, if this access would result in behavior that depends on unauthenticated or inconsistent data**

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 **a failure occurs during any operation used to store persistent objects (data and keys)**.

Application Note:

This SFR enforces atomicity of any write operation [GPIAPI], [TR-DEVAPI].

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 role** to specify alternative initial values to override the default values when an object or information is created.

8.1.7 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.

8.1.8 Random Number Generator

FCS_RNG.1 Random numbers generation

FCS_RNG.1.1 The TSF shall provide a **hybrid** random number generator that implements: **none**.

FCS_RNG.1.2 The TSF shall provide random numbers that meet **NIST SP 800-90A**.

Application Note:

A hardware RNG is used as an entropy source (the nature is SoC-dependent).

8.2 Security Assurance Requirements

The Security Target introduces the EAL TEE, which consists of the predefined assurance package EAL2 where AVA_VAN.2 (Vulnerability analysis) is refined to increase the attack potential from Basic to Enhanced-Basic.

8.3 Security Requirements Rationale

8.3.1 Objectives

8.3.1.1 Security Objectives for the TOE

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

- FIA_ATD.1 enforces the management of the user identity as security attribute, which then become TSF data, protected in integrity and confidentiality
- FIA_UID.2 requires the identification of any user before any action, thus allowing the access to services and data to authorized users only.
- FIA_USB.1 enforces the association of the user identity to the active entity that acts on behalf of the user and to check that this is a valid identity

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

- FAU_SAR.1 enforces TEE identifier access capabilities
- FCS_RNG.1 enforces statistical uniqueness of the TEE identification data if it is generated on the TOE.

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_USB.1 and FIA_UID.2 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 space

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, FMT_MSA.1/Runtime, FMT_MSA.3/Runtime and FDP_IFF.1/Runtime ensure read TEE and TA runtime data policy, which grants read access to authorized entities only
- FPT_ITT.1/Runtime ensures protection against disclosure of TEE and TA data that is transferred between resources
- FDP_RIP.1/Runtime states resource clean up policy.

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

- FDP_IFC.2/Runtime, FMT_MSA.1/Runtime, FMT_MSA.3/Runtime and FDP_IFF.1/Runtime state TEE and TA runtime data policy, which grants write access to authorized entities only
- FPT_ITT.1/Runtime ensures protection against modification of TEE and TA data that is transferred between resources.

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

- FDP_SDI.2 enforces the consistency and authenticity of TA code
- 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, FMT_MSA.1/Runtime, FMT_MSA.3/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:

- FDP_SDI.2 monitors the authenticity and consistency of TEE persistent data and states the response upon failure
- FCS_COP.1 states the cryptography used to protect consistency, authenticity and confidentiality of the TEE data in external memory, if applicable
- 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, FMT_MSA.1/Runtime, FMT_MSA.3/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 the 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.

8.3.2 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.TEE_ID	FAU_SAR.1, FCS_RNG.1, FAU_STG.1	Section 4.3.1
O.INSTANCE_TIME	FPT_STM.1/Instance time	Section 4.3.1
O.OPERATION	FDP_IFC.2/Runtime, FAU_ARP.1, FDP_SDI.2, FIA_ATD.1, FIA_UID.2, FIA_USB.1, FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FMT_SMF.1 FMT_SMR.1, FPT_FLS.1	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_RIP.1/Runtime, FPT_ITT.1/Runtime, FMT_MSA.1/Runtime, FMT_MSA.3/Runtime	Section 4.3.1
O.RUNTIME_INTEGRITY	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FPT_ITT.1/Runtime, FMT_MSA.1/Runtime, FMT_MSA.3/Runtime	Section 4.3.1

Security Objectives	Security Functional Requirements	Rationale
O.TA_ISOLATION	FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FCS_COP.1, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FMT_SMF.1, FPT_FLS.1, FMT_MSA.1/Runtime, FMT_MSA.3/Runtime	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, FMT_MSA.1/Runtime, FMT_MSA.3/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, FDP_ITT.1/Trusted Storage, FPT_FLS.1	Section 4.3.1
O.TA_AUTHENTICITY	FDP_SDI.2	Section 4.3.1

Table 18 Security Objectives and SFRs - Coverage

Security Functional Requirements	Security Objectives	Rationale
FIA_UID.2	O.CA_TA_IDENTIFICATION, O.OPERATION	
FIA_USB.1	O.CA_TA_IDENTIFICATION, O.OPERATION	
FMT_SMR.1	O.OPERATION	
FIA_ATD.1	O.CA_TA_IDENTIFICATION, O.OPERATION	
FAU_SAR.1	O.TEE_ID	
FAU_STG.1	O.TEE_ID	
FDP_RIP.1/Runtime	O.RUNTIME_CONFIDENTIALITY	
FMT_MSA.1/Runtime	O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION	
FMT_MSA.3/Runtime	O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION	
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	
FPT_ITT.1/Runtime	O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION	

Security Functional Requirements	Security Objectives	Rationale
FCS_COP.1	O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE	
FDP_SDI.2	O.OPERATION, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, O.TA_AUTHENTICITY	
FPT_INI.1	O.TRUSTED_STORAGE	
FAU_ARP.1	O.OPERATION	
FPT_FLS.1	O.OPERATION	
FMT_SMF.1	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE	
FPT_TEE.1	O.TA_AUTHENTICITY	
FDP_ACC.1/Trusted Storage	O.TA_ISOLATION, O.TRUSTED_STORAGE	
FDP_ACF.1/Trusted Storage	O.TA_ISOLATION, O.TRUSTED_STORAGE	
FDP_ROL.1/Trusted Storage	O.TRUSTED_STORAGE	
FMT_MSA.1/Trusted Storage	O.TA_ISOLATION, O.TRUSTED_STORAGE	
FMT_MSA.3/Trusted Storage	O.TA_ISOLATION, O.TRUSTED_STORAGE	
FCS_RNG.1	O.TEE_ID, O.RNG	
FPT_STM.1/Instance time	O.INSTANCE_TIME	

Table 19 SFRs and Security Objectives

8.3.3 Dependencies

8.3.3.1 SFRs Dependencies

Requirements	CC Dependencies	Satisfied Dependencies
FIA_UID.2	No Dependencies	
FMT_SMR.1	(FIA_UID.1)	FIA_UID.2
FIA_USB.1	(FIA_ATD.1)	FIA_ATD.1
FIA_ATD.1	No Dependencies	
FAU_SAR.1	(FAU_GEN.1)	
FAU_STG.1	(FAU_GEN.1)	
FMT_SMF.1	No Dependencies	
FPT_TEE.1	No Dependencies	
FDP_RIP.1/Runtime	No Dependencies	

Requirements	CC Dependencies	Satisfied Dependencies
FMT_MSA.1/Runtime	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FDP_IFC.2/Runtime
FMT_MSA.3/Runtime	(FMT_MSA.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_MSA.1/Runtime
FDP_IFC.2/Runtime	(FDP_IFF.1)	FDP_IFF.1/Runtime
FDP_IFF.1/Runtime	(FDP_IFC.1) and (FMT_MSA.3)	FMT_MSA.3/Runtime, FDP_IFC.2/Runtime
FPT_ITT.1/Runtime	No Dependencies	
FCS_CKM.1	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_CKM.4, FCS_COP.1
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	FCS_CKM.1
FCS_COP.1	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1, FCS_CKM.4
FDP_SDI.2	No Dependencies	
FPT_INI.1	No Dependencies	
FAU_ARP.1	(FAU_SAA.1)	
FPT_FLS.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, 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
FCS_RNG.1	No Dependencies	
FPT_STM.1/Instance time	No Dependencies	

Table 20 SFRs Dependencies

8.3.3.1.1 Rationale for the exclusion of Dependencies

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 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.

The dependency FMT_SMF.1 of FMT_MSA.1/Runtime is discarded. This ST does not define any management function of the security attributes defined in FMT_MSA.1/Runtime.

The dependency FMT_SMF.1 of FMT_MSA.1/Trusted Storage is discarded. This ST does not define any management function of the security attributes defined in FMT_MSA.1/Trusted Storage.

8.3.3.2 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	

Requirements	CC Dependencies	Satisfied 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 21 SARs Dependencies

8.3.4 Rationale for the Security Assurance Requirements

The assurance level defined in this Security Target following [PP-TEE] 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 of [PP-TEE].

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 of [PP-TEE].

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.

8.3.5 AVA_TEE.2 TEE vulnerability analysis

This ST provides the set of Security Assurance Requirements (SARs) which consists of the EAL 2 package augmented with the extended AVA_TEE.2 SAR, which refines the AVA_VAN.2 Basic attack potential to a TEE-Low attack potential.

9 TOE SUMMARY SPECIFICATION

9.1 TOE Summary Specification

This section contains the definition and description of the TOE summary specification (TSS) that meets the security functional requirements.

Cryptographic Support

There are several functions within the TOE related to cryptographic support: generation of random numbers, digital signature (generation and verification), data encryption and decryption, key destruction, the generation of hash values and the generation and verification of MAC values.

- generation of cryptographic keys in accordance with the cryptographic key generation algorithms key sizes specified under FCS_CKM.1
- destruction of cryptographic keys by erasure of volatile memory areas containing cryptographic keys or overwriting value of key encryption keys
- hash calculation in accordance with the cryptographic algorithms specified under FCS_COP.1
- HMAC calculation and verification in accordance with the cryptographic algorithms specified under FCS_COP.1
- signature generation and signature verification in accordance with the cryptographic algorithms specified under FCS_COP.1
- encryption and decryption in accordance with the cryptographic algorithms specified under FCS_COP.1
- random number generator that meet NIST SP 800-90A.

TA_CA_Identification

All TAs or CAs are assigned a unique identifier.

- The TOE identifies the application before performing any further actions.
- The TOE supports the management of the user identity as security attribute, which then become TSF data, protected in integrity and confidentiality.
- The TOE allows access to services and data to authorized users only. The identification of any user before any action is mandatory.
- The TOE associate user identity security attributes with subjects acting on the behalf of that user.
- The TOE provides the roles: TSF, TA_User and associates users with roles.

Device Identification

The TOE provides a service for retrieving the unique device identifier. No service is provided for storing it. The unique device ID is written in an audit record by the Trusted Firmware, stored in persistent memory, and is accessible only to authorized users (Trusted Applications (TA), with security attribute "TA_identity", "TA_properties").

Operation and Firmware Integrity

The Trusted Firmware ensures the integrity of the TOE. The TOE ensures that security services are invoked and completed before each TOE service is allowed to proceed. Also, a security domain is maintained to protect the execution of TOE's services from interference and tampering by untrusted subjects.

Secure services are provided to ensure the return of the TOE to a secure state after failures or service discontinuities.

Secure Initialization

The TOE provides an initialization process ensuring that all security features are initialized in a secure way during the start-up. The initialization process guarantees the TSF integrity from the power-off state into an initial secure state. The initialization process includes the chipset initialization (TOE Environment: A.Power-Up, OE.INITIALIZATION) and the Kinibi initialization.

The TOE reaches a secure state after a successful completion of the initialization sequence in proper order.

Trusted Application Authentication

The TOE verifies all TA code prior to execution.

Runtime Data Information Flow Control

Isolated Execution: ensures TAs execute completely isolated from and unhindered by others and guarantees that any code and data is protected at runtime.

Integrity: The TOE monitors the integrity at runtime of all data objects it stores and exchanges, including cryptographic keys, cryptographic parameters, TA data, security attributes and other security critical information.

Confidentiality: The TOE provides the security services to ensure that the following security-critical assets are protected against unauthorized disclosure and/or modification as appropriate:

- data belonging to the TA or to the TEE itself (Stands for parameter values, return values, content of memory regions in cleartext)
- Implementation and data necessary to provide security services.

Trusted Communication

The TOE ensures that data transmitted between itself and the users (TA, REE, Trusted Firmware) is protected against unauthorized disclosure and modification including deletion, insertion and replay.

The TOE provides a channel for the user to input data to the TEE and for the TEE to output data to the user; the channel protects against eavesdropping and tampering.

Trusted Storage

The TOE protects persistently stored data (e.g. cryptographic keys) belonging to a certain TA from being accessed by other TAs.

- The TOE enforces the different security function policies on subjects (S.API), objects (OB.TA_STORAGE) and operations (OP.LOAD, OP.STORE) based on different security attributes (S.API.caller, OB.TA_STORAGE.owner and OB.TA_STORAGE.inExtMem). Any processing is only allowed if the respective Trusted Storage security attribute has the correct value.

- The management of the Trusted Storage security attributes are restricted to different roles and based on different rules. The TOE checks if there are no restrictions violated before processing the management of the Trusted Storage security attribute.
- The TOE ensures that only secure values are accepted for Trusted Storage security attributes. The TOE supports the static security attribute initialization. Different security enforcing policies are allowed to provide permissive and/or restrictive default values for Trusted Storage security attributes.

9.2 SFRs and TSS

9.2.1 SFRs and TSS - Rationale

9.2.1.1.1 Identification

FIA_UID.2 This SFR is fully covered by TA_CA_Identification covering the access control to services and data by authorized users only.

FMT_SMR.1 This SFR is fully covered by TA_CA_Identification covering the management of roles: TSF, TA_User.

FIA_ATD.1 This SFR is fully covered by TA_CA_Identification covering the management of the user identity.

FAU_STG.1 This SFR is fully covered by Runtime Data Information Flow Control covering the access control to TEE execution space.

FAU_SAR.1 This SFR is fully covered by Device Identification covering the capability to read Device ID by TA from the audit records and to interpret the information.

9.2.1.1.2 Confidentiality, Integrity and Isolation

FDP_RIP.1/Runtime This SFR is fully covered by Runtime Data Information Flow Control covering the management of previous information content and the resource clean up policy.

FMT_MSA.1/Runtime This SFR is fully covered by Runtime Data Information Flow Control covering the access control to TA execution space and the TEE and TA runtime data policy, which grants R/W access to authorized entities only.

FMT_MSA.3/Runtime This SFR is fully covered by Runtime Data Information Flow Control covering the access control to TA execution space and the TEE and TA runtime data policy, which grants R/W access to authorized entities only.

FDP_IFC.2/Runtime This SFR is fully covered by Runtime Data Information Flow Control covering the access control to TA execution space and the TEE and TA runtime data policy, which grants R/W access to authorized entities only.

FDP_IFF.1/Runtime This SFR is fully covered by Runtime Data Information Flow Control covering the access control to TA execution space and the TEE and TA runtime data policy, which grants R/W access to authorized entities only.

FPT_ITT.1/Runtime This SFR is fully covered by Runtime Data Information Flow Control and Trusted Communication covering the protection against disclosure of TEE and TA data that is transferred between resources.

9.2.1.1.3 Cryptography

FCS_CKM.1 This SFR is fully covered by Cryptographic Support covering Key Generation.

FCS_CKM.4 This SFR is fully covered by Cryptographic Support covering Key Destruction.

FCS_COP.1 This SFR is fully covered by Cryptographic Support covering cryptographic operation.

9.2.1.1.4 Initialization, Operation and Firmware Integrity

FDP_SDI.2 This SFR is fully covered by Operation and Firmware Integrity covering the monitoring of user data stored and the actions upon detection of a data integrity error.

FPT_INI.1 This SFR is fully covered by Secure Initialization covering the TOE secure initialization function.

FAU_ARP.1 This SFR is fully covered by Operation and Firmware Integrity covering actions upon detection of potential security violation.

FPT_TEE.1 This SFR is fully covered by Trusted Application Authentication covering TA instantiation.

FPT_FLS.1

- This SFR is fully covered by Secure Initialization covering the preservation of a secure state upon initialization or device binding failure.
- This SFR is fully covered by Operation and Firmware Integrity covering the management of failures that result in non-secure state.

9.2.1.1.5 Trusted Storage

FDP_ACC.1/Trusted Storage This SFR is fully covered by Trusted Storage covering the management of Trusted Storage Access Control.

FDP_ACF.1/Trusted Storage This SFR is fully covered by Trusted Storage covering the management of Trusted Storage Access Control.

FDP_ROL.1/Trusted Storage This SFR is fully covered by Trusted Storage covering the Trusted Storage Access Control to permit the rollback of write operations (atomicity).

FMT_MSA.1/Trusted Storage This SFR is fully covered by Trusted Storage covering the Trusted Storage Access Control and the permission/restriction management of Trusted Storage security attributes.

FMT_MSA.3/Trusted Storage This SFR is fully covered by Trusted Storage covering the Trusted Storage Access Control and the permission/restriction management of Trusted Storage security attributes.

9.2.1.1.6 Random Number Generator

FCS_RNG.1 This SFR is fully covered by Cryptographic Support covering Random Number generation that meet NIST SP 800-90A.

9.2.1.2 TOE Summary Specification

Operation and Firmware Integrity

- The TOE prevents and reacts from potential security violation. The covered security functional requirement is FAU_ARP.1.
- The TOE applies a cryptographic protocol (hash) to ensure that data integrity is controlled by the TSF. The covered security functional requirement is FDP_SDI.2.
- The TOE prevents failures that result in non-secure state and maintains a secure state upon initialization or device binding failure. The covered security functional requirement is FPT_FLS.1.

Secure Initialization

- The TOE prevents failures that result in non-secure state and maintains a secure state upon initialization or device binding failure. The covered security functional requirement is FPT_FLS.1.

9.2.2 Association tables of SFRs and TSS

Security Functional Requirements	TOE Summary Specification
FIA_UID.2	TA_CA_Identification
FMT_SMR.1	TA_CA_Identification
FIA_ATD.1	TA_CA_Identification

Security Functional Requirements	TOE Summary Specification
FAU_SAR.1	Device Identification
FAU_STG.1	Runtime Data Information Flow Control
FDP_RIP.1/Runtime	Runtime Data Information Flow Control
FMT_MSA.1/Runtime	Runtime Data Information Flow Control
FMT_MSA.3/Runtime	Runtime Data Information Flow Control
FDP_IFC.2/Runtime	Runtime Data Information Flow Control
FDP_IFF.1/Runtime	Runtime Data Information Flow Control
FPT_ITT.1/Runtime	Runtime Data Information Flow Control, Trusted Communication
FCS_CKM.1	Cryptographic Support
FCS_CKM.4	Cryptographic Support
FCS_COP.1	Cryptographic Support
FDP_SDI.2	Operation and Firmware Integrity
FPT_INI.1	Secure Initialization
FAU_ARP.1	Operation and Firmware Integrity
FPT_TEE.1	Trusted Application Authentication
FPT_FLS.1	Operation and Firmware Integrity, Secure Initialization
FDP_ACC.1/Trusted Storage	Trusted Storage
FDP_ACF.1/Trusted Storage	Trusted Storage
FDP_ROL.1/Trusted Storage	Trusted Storage
FMT_MSA.1/Trusted Storage	Trusted Storage
FMT_MSA.3/Trusted Storage	Trusted Storage
FCS_RNG.1	Cryptographic Support

Table 22 SFRs and TSS - Coverage

TOE Summary Specification	Security Functional Requirements
Cryptographic Support	FCS_CKM.1, FCS_CKM.4, FCS_COP.1, FCS_RNG.1
TA_CA_Identification	FIA_UID.2, FMT_SMR.1, FIA_ATD.1
Device Identification	FAU_SAR.1
Operation and Firmware Integrity	FDP_SDI.2, FAU_ARP.1, FPT_FLS.1
Secure Initialization	FPT_INI.1, FPT_FLS.1
Trusted Application Authentication	FPT_TEE.1

TOE Summary Specification	Security Functional Requirements
Runtime Data Information Flow Control	FDP_RIP.1/Runtime, FMT_MSA.1/Runtime, FMT_MSA.3/Runtime, FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FPT_ITT.1/Runtime, FAU_STG.1
Trusted Communication	FPT_ITT.1/Runtime
Trusted Storage	FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage

Table 23 TSS and SFRs - Coverage