Security Target

Kinibi v510A Security Target

TRUSTONIC

PREFACE

TABLE OF CONTENTS

1 INTRODUCTION

This document is the Security Target for Kinibi 510A 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 automotive ECU such as IVI or instrument cluster systems that enables security services such as:

- In-Car Payments,
- Vehicle ECU secure update management,
- Engine mapping update management
- Secure engine sensors remote logging

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 Audience

During the evaluation: Trustonic employees, Riscure evaluation lab, NSCIB evaluation scheme

General publication once the evaluation is completed.

1.2 Revision History

Date **[day month year]**	**Version**	**Description**
28 February 2022	1.2	First public version

Table 1-1: Revision History

2 IDENTIFICATION

2.1 ST Identification

Security Target Identification	
Document title	Kinibi 510A Security Target
Document reference	TT_510A_CC_ST
Document version	1.2
Document publication date	28 February 2022
Document status	*Final / Public Release*
Document author	*TRUSTONIC*

2.2 TOE Identification

2.2.1 TOE Type

TOE characterization	
TOE Type	*TEE-Part (TEE Trusted OS)*
Multiple TOEs	*No*

2.2.2 TOE References

TEE Trusted OS identification	
Developer	TRUSTONIC
Commercial name	KINIBI
Product version	Kinibi-510a-V007
Source[1]	Kinibi-Src-510a-V007 `fc5c8d3cd4c0ec29e88820769cb3e6269116114af785eb08f6cd9377083d316e Kinibi-Src-Exynos64-510a-V007-r0-20211117.125042-1-20211117_124837_19.zip`

[1] Source delivery contains platform specific code that might require modification during SoC integration stage. Platform specific code can be found in dedicated folders *`Kinibi\MTK \Locals\Code\arm\arm{32,64}`* and *`Drivers\DrRPMB\Locals\Code\platforms`*)

2.3 Non-TOE Components Identification

Non-TOE SoC identification (Reference board / used for test purposes)	
Developer	96boards
Commercial name	HiKey960
Chipset Identifiers	Kirin 960 SoC
Reference of Chipset	Kirin 960 SoC
Version of Chipset	Kirin 960 SoC
ROM code identifier	Kirin 960 SoC
Boot code identifier	HiSilicon hisi-sec_xloader.img (Dec 6 2017)
Secure Monitor identifier	ARM Trusted Firmware-A v1.5
Product package version[2]	t-base-HIKEY960-Android-510a-V007
Product package file name	t-base-HIKEY960-Android-510a-V007-20211117_105751_32841.zip
Self-identification	Via tlApiGetMobiCoreVersion(), see AGD_PRE.1.1C: t-base-HIKEY960-Android-510a-V007-20211117_105751_32841_0d92e

Image hashes (Sha256)	`0d70adcf617b2158893dde281aa8e12a2278ad6091023c7fd505839f2a682c5d`
	`Hikey960/recovery/bin/hisi-sec_xloader.img`
	`e2949f903c1044da3e74bb086af12eb7a8cb53013f588fbd637083e0476e2f94`
	`./Hikey960/boot_images/HUAWEI_AARCH64_HIKEY960/Release/bl2.bin`
	`5fb9d3a3e951a376f3052f6fe200585d2b6e15da6e463e63564d9a8047ae71f2`
	`./Hikey960/boot_images/HUAWEI_AARCH64_HIKEY960/Release/fip.bin`
	
	`fip.bin includes:`
	`bd1d628405a422c28ec8861b30723e65fe497e0e1b3a703f0754f731de54969e`
	`./Hikey960/boot_images/HUAWEI_AARCH64_HIKEY960/Release/lpm3.img`
	`e6693ea2efbafaee8f7e7470b6af328f12af79b4e618abf24c45b691ef068cfb`
	`./Hikey960/boot_images/HUAWEI_AARCH64_HIKEY960/Release/bl31.bin`
	`a501dfaddacc120a0050ec88558ea6770d19bf1ceaecb4d799ee80ae517dea5a`
	`/Hikey960/boot_images/HUAWEI_AARCH64_HIKEY960/Release/BL33_AP_UEFI.fd`
	`83abaddf9cee26aa7e8ef7b8f6b3263be8231ebbf3ae9fc093bd6a0b7b3931e0`
	`./SecureIntegration/.../Bin/MobiCore/HUAWEI_AARCH64_HIKEY960/Release/tee.img`

[2] This is the Product package provided to the evaluation lab for testing purpose. It is a Kinibi OS build dedicated for the reference board being the Hikey 960. The TOE itself is meant to be adapted to any platform where it needs to run.

Non-TOE software components identification			
Name	**Reference**	**Main developer**	**SoC/Device reference**
Rich OS: Android	Android P	Google	
Kinibi android kernel driver	510A	Trustonic	
Kinibi android daemon	510A	Trustonic	
Client API	510A	Trustonic	

2.4 Security Guidance Identification

Guidance for SoC integrators	
Document title	Kinibi Integration Guide
Document reference	Kinibi_Integration_Guide.pdf
Document version	510A
Document publication date	November 10, 2020
Document status	*Final / Confidential*
Document author	*Trustonic*

Guidance for SoC integrators	
Document title	Kinibi Driver Developers Guide
Document reference	Kinibi_Driver_Developers_Guide.pdf
Document version	2.9
Document publication date	March 18th, 2020
Document status	*Final / Confidential*
Document author	*Trustonic*

Guidance for SoC integrators	
Document title	Kinibi Driver API Documentation
Document reference	Kinibi_Driver_API_Documentation.pdf
Document version	2.14
Document publication date	August 25th , 2020

Guidance for SoC integrators and TA developers	
Document title	Kinibi Developers Guide
Document reference	Kinibi_Developers_Guide.pdf

Guidance for SoC integrators and TA developers	
Document version	5.5
Document publication date	August 31st, 2020

Guidance for SoC integrators and TA developers	
Document title	Kinibi API Documentation
Document reference	Kinibi_API_Documentation.pdf
Document version	API Level 12
Document publication date	November 18th, 2020

Guidance for SoC integrators	
Document title	Kinibi v510A Preparative Procedures Guidance
Document reference	Trustonic-Kinibi-AGD_PRE_1.13.pdf
Document version	1.13
Document publication date	January 20, 2022

Guidance for SoC integrators and TA developers	
Document title	Kinibi v510A Operational User Guidance
Document reference	Trustonic-Kinibi-AGD_OPE_1.9.pdf
Document version	1.9
Document publication date	September 8, 2021

2.5 TOE Configuration

TOE configuration: Once delivered, Kinibi OS image shall be configured (binary image patching, no need to recompile Kinibi OS) through Trustonic provided tools by the OEM. For this ST, we consider two modes:

- **Normal mode**: This mode corresponds to the GP TEE base PP. Kinibi Secure Storage is not rollback protected and Trusted Application are not downgrade protected.
- **Time&RollbackProtection (T&R) mode:** Secure Storage Rollback Protection and Trusted Application downgrade protection are enforced with RPMB support. In this configuration, the TOE enforces GP TEE Time & Rollback PP modules security objectives.

Important to note that Trusted Application downgrade can be enabled without Trusted Storage rollback protection as soon as the platform provides RPMB service. This mode is not considered in this ST for simplicity.

2.6 Developers and Manufacturers Identification.

Developer/ Manufacturer Company Name	Legal Address	Contact	TOE-related sites	Site audits/date
Trustonic	*Les Aqueducs Bâtiment 2* *535 route des Lucioles* *Sophia Antipolis* *06560 Valbonne* *France*	*Lukas Hänel*	*Sophia Antipolis (FR)*	*Nov 2016: Sophia site has been audited by Thales in the context of Kinibi 311 CC evaluation (ALC_DEL).*

3 CONFORMANCE CLAIMS

3.1 CC Conformance Claim

This Security Target conforms to CC Part 2 extended and Part 3 conformant, with a claimed Evaluation Assurance Level of EAL 5, augmented by ALC_FLR.1.

This Security Target claims conformance to the following specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Requirements, Version 3.1, Revision 5, April 2017.
- Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Requirements, Version 3.1, Revision 5, April 2017; augmented by ALC FLR.1.

3.2 PP Claim

This Security Target does not claim conformance to any PP.

3.3 TEE PP Vs. Kinibi ST

This ST has been derived from the GlobalPlatform TEE Protection Profile but differs from it.

- The TOE scope in this ST is limited to the Trusted OS while the TOE in [PP-TEE] covers Hardware, Secure Firmware and Trusted OS. The Hardware and Secure Firmware parts are considered as environment of this TOE, covered by security objectives for the environment.
- The TOE threats in this ST are limited to attacks through either wireless or standard physical connection of other equipment, to the vehicle systems, attacks from vehicle system software outside of the TEE.

These differences cause changes in this ST regarding the TOE in [PP-TEE]. These changes affect:

1. Threats: Threat model adapted to automotive use case.
2. SFRs
   a. SFRs that are fulfilled by the Chipset disappear from the current ST.
   b. SFRs that are fulfilled by both SW and HW parts are modified in the current ST to cover only the SW part.
3. New Assumptions or Objectives for environment are added to environmentally protect the TOE and cover the security of HW/FW parts.
4. Objectives for the TOE (O)
   a. Objectives for the TOE in the PP TEE 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		×

TOE SPDs	PP TEE	Included
A.PEER.MGT		×
A.RNG		×
A.POWER_UP		×
A.CAR_PHYSICAL_PROTECTION		×
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_DOWNGRADE	×	×
T.STORAGE_CORRUPTION	×	×
T.ROLLBACK	×	×
T.TA_PERSISTENT_TIME_ROLLBACK	×	×
Organizational Security Policies		
OSP.INTEGRATION_CONFIGURATION	×	×
OSP.SECRETS	×	×
OSP.TEE_ID		×
OSP.TA_MANAGEMENT		×

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	×	×

TOE Objectives	PP TEE	Included
O.KEYS_USAGE	×	×
O.TEE_ID	×	×
O.INITIALIZATION	×	changed into OE. INITIALIZATION
O.INSTANCE_TIME	×	×
O.OPERATION	×	×
O.RNG	×	×
O.RUNTIME_CONFIDENTIALITY	×	×
O.RUNTIME_INTEGRITY	×	×
O.TA_AUTHENTICITY	×	×
O.TA_ISOLATION	×	×
O.TEE_DATA_PROTECTION	×	×
O.TEE_ISOLATION	×	×
O.TRUSTED_STORAGE	×	×
O.ROLLBACK_PROTECTION	×	×
O.TA_PERSISTENT_TIME	×	×
O.ROLLBACK_PROTECTION $^{(T\&R)}$	×	×
O.PERSISTENT_TIME $^{(T\&R)}$	×	×
Security Objectives for the Operational Environment		
OE.INTEGRATION_CONFIGURATION	×	×
OE.PROTECTION_AFTER_DELIVERY	×	×
OE.ROLLBACK	×	×
OE.SECRETS	×	×
OE.TA_DEVELOPMENT	×	×
OE.CONFIGURATION		×
OE.TRUSTED_HARDWARE		×
OE.TRUSTED_FIRMWARE		×
OE.UNIQUE_TEE_ID		×

TOE Objectives	PP TEE	Included
OE.TA_MANAGEMENT		×
OE.RNG		×
OE.INITIALIZATION		×
OE.CAR_PHYSICAL_PROTECTION		×

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)	Non-TOE HW/FW/SW
FIA_ATD.1	×	
FIA_UID.2	×	
FIA_USB.1	×	
FMT_SMR.1	×	
FDP_IFC.2/Runtime	×	Underlying Hardware and Secure Firmware must provide correct MMU operation and correct DRAM isolation configuration. (OE.TRUSTED_HARDWARE ; OE.TRUSTED_FIRMWARE)
FDP_IFF.1/Runtime	×	same remark as FDP_IFC.2/Runtime
FDP_ITT.1/Runtime		Removed. Non-TOE HW only
FDP_RIP.1/Runtime	×	
FPT_ITT.1/Runtime	×	
FCS_COP.1	×	
FDP_ACC.1/TA_keys	×	
FDP_ACF.1/TA_keys	×	
FMT_MSA.1/TA_keys	×	
FMT_MSA.3/TA_keys	×	
FAU_ARP.1	×	Hardware is responsible for aborts on invalid memory accesses (OE.TRUSTED_HARDWARE)
FDP_SDI.2	×	
FPT_FLS.1	×	TOE Environment: Platform Hardware and Secure Firmware is assumed to respond securely to operation failures such as Platform and Secure Firmware initialization, TEE OS integrity or downgrade check failure, invalid command handling in the Secure Firmware and Hardware operation failure.

TOE SFRs (in PP TEE)	**Kinibi (SFRs included in this ST)**	**Non-TOE HW/FW/SW**
FPT_INI.1		Removed : the TOE is loaded by a firmware bootloader
FMT_SMF.1	×	
FPT_TEE.1	×	
FAU_SAR.1	×	shared responsibility between SW and HW: retrieved or computed in the TOE from hw-protected data
FAU_STG.1		Removed: Non-TOE HW only. TEE Identifier generation, storage on device and access API for the TEE OS is covered in OE.UNIQUE_TEE_ID.
FPT_STM.1/Instance time	×	
FCS_RNG.1	×	combination: hardware entropy source + crypto PRNG. Hardware RNG support is covered by OE.RNG.
FDP_ACC.1/Trusted Storage	×	Kinibi Trusted storage relies on a device specific secret key. Provisioning of this secret is covered by OE.SECRETS. Access to this secret and derivation into a Storage Root of Trust is platform specific and handled during the phase 2 of the TOE lifecycle: “Kinibi porting for a specific Silicon Provider SoC”
FDP_ACF.1/Trusted Storage	×	
FDP_ROL.1/Trusted Storage	×	
FMT_MSA.1/Trusted Storage	×	
FMT_MSA.3/Trusted Storage	×	
FDP_ITT.1/Trusted Storage		Removed: Non-TOE HW only

TOE SFRs (in PP TEE)	Kinibi (SFRs included in this ST)	Non-TOE HW/FW/SW
FDP_SDI.2/Rollback$^{(T\&R)}$	×	Kinibi leverages RPMB flash storage to enforce rollback protection on the Trusted Storage. RPMB service is offered to the Kinibi Trusted OS through a secure driver developed by the SIP with the help of Trustonic during the phase 3 of the TOE lifecycle: “SIP and OEM integration”.
FPT_FLS.1/Rollback$^{(T\&R)}$	×	
FPT_STM.1/Persistent Time$^{(T\&R)}$	×	
FMT_MTD.1/Persistent Time$^{(T\&R)}$	×	
FMT_SMF.1/Persistent Time$^{(T\&R)}$	×	

Table 3 PP SFRs vs. ST

4 TOE Description

4.1 Overview

The TOE is the Kinibi Secure OS 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.

Kinibi 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 that has processing capabilities, 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. [TEE SA 1.2]

Kinibi leverages ARM Exception Levels and the MMU to keep the separation between the TEE OS and the TAs and between the TAs themselves. Kinibi leverages the ARM TrustZone to split the platform in two distinct areas, the Normal World with a conventional rich operation system and rich applications and the Secure World.

Configuration of REE/TEE hardware isolation (DRAM, peripherals) and secure loading of Kinibi OS in a dedicated secure memory area is secure firmware responsibility (covered in OE.TRUSTED_HARDWARE and OE.TRUSTED_FIRMWARE).

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 Secure Drivers, Trusted Applications with extra rights, that run on Kinibi and are allowed to 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 above Client API’s 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 Secure Drivers 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

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 Secure Drivers (SD) that are not embedded into Kinibi image
- The Rich Execution Environment
- The Client Applications CA
- The Hardware and secure firmware device platform.

In the following, TOE and Kinibi are used interchangeably.

SiP/OEM-signed privileged Trusted Applications and Secure Drivers can be loaded in the field, thus giving rise to modified versions of Kinibi. To maintain the security level as defined in this ST, the same evaluation criteria should be applied to the new components and resulting Kinibi versions.

TOE operation modes: Kinibi OS only support one operation mode: **production mode**.

4.2 Expected Usage

The purpose of the TOE 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.

The following sections define the TOE security functionalities and the TOE expected usage.

Security features provided by the Trusted Applications are out of the scope of the TOE.

4.2.1 Kinibi Security Features

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

4.2.1.1 TOE components authenticity and rollback protection

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

Kinibi relies on hardware roots of trust (secure boot) for authenticity and rollback protection of Kinibi image.

Kinibi then ensures the authenticity of the SIPs/OEMs Trusted Applications running in the Secure World.

Downgrade protection of the SIPs/OEMs Trusted Applications and Secure driver can be enforced in two ways depending on the platform integration:

- through a rotation of TAs and Secure Drivers signing key embedded in TEE image (platform integrator responsibility to rollout a new version of the TEE image also protected against downgrade).
- through a storage of TAs and Secure Drivers version in a RPMB space when such security service is integrated. Version storage and verification is performed by Kinibi, but TA version upgrade and downgrade protection configuration is SIP/OEM responsibility.

4.2.1.2 Memory Management and Isolation mechanism

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 2 memory regions in its bootloader arguments:

1. *Secure DRAM:* The memory region is inaccessible from NWd.
2. *Non-Secure DRAM:* The memory region is accessible from NWd.

Kinibi ensures:

- Isolation of the Kinibi OS and all the TAs and SDs from the REE.
- Isolation between TAs and isolation of the Kinibi OS and its SDs from TAs.

Kinibi implements several mechanisms to achieve isolation:

- CPU register saving and restoring on context switches.
- Virtual memory to ensure TAs cannot access arbitrary memory.
- Restricted memory management via the TA API.

Using these isolation mechanisms, Kinibi guarantees the correct execution of SD and TA code.

Note: Secure Drivers are Trusted Application with extra rights, that can access arbitrary memory locations. To keep the correct execution guarantee of the TEE OS, Secure Drivers developers must follow driver development rules described in Kinibi Driver Developer Guide.

Note: *Kinibi OS is not responsible for the configuration of Secure DRAM and Non-Secure DRAM memory ranges (e.g. ARM TrustZone firewall configuration).*

4.2.1.3 User Identification

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

Kinibi is able to load 3 types of Trusted Applications:

- Embedded TAs are embedded inside the Kinibi image. Owner of these TAs is the owner of Kinibi image signing key (SIP).
- System TAs are stored in the NWd filesystem. Must be signed by a private key owned by the SIP or the OEM, or by a Trustonic private key.
- SP TAs are stored in the Secure Filesystem and delivered through an Over-The-Air protocol.

Kinibi Trusted Applications can be Proprietary TAs which uses legacy APIs or Global-Platform-type TAs which uses GP APIs.

Drivers and Middleware (collectively called Secure Drivers) are developed by Trustonic, by the SIP and by the OEMs.

The identity of Trusted Application is composed of their type (Embedded, System, SP), their origin (SP-TAs) and the Trusted Application specific UUID.

When a Trusted Application is invoked, the origin of the call is securely provided by Kinibi. Following is a list of possible origins:

- Call from a CA in REE. The identity of a Client Application in REE is ensured by the REE. Kinibi 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.
- Call from the Kinibi OS.
- Call from another Trusted Application identified by its unique identity.

4.2.1.4 Inter-TA communication

Kinibi offers communication primitives with memory sharing between GP TAs. These securely controlled primitives ensure isolation of memory between participating TAs.

4.2.1.5 Secure storage of Keys and Data

Kinibi offers a mechanism to wrap data into Secure Objects.

- Data is stored 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.
- Secure Object are bound to the TA identity and to the device**.** Every Secure Object is encrypted with a key derived from the device master key 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 entity except Kinibi on the device that has generated it.

Kinibi also offers a Trusted storage service for TA data and keys and Kinibi OS data and keys which ensures:

- Confidentiality, authenticity and consistency of each TA stored data and keys
- Atomicity of the operations that modify the storage
- Device binding

In the Time&RollbackProtection (T&R) mode: Kinibi also enforces Rollback protection on Trusted storage (Secure objects are not rollback protected).

4.2.1.6 Key Management and Cryptographic operations

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

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.

4.2.1.6.1 GlobalPlatform API

The following tables list the cryptographic algorithms that Kinibi-510A supports:

GP Algorithm name (TEE_TYPE_*)	Possible Key Sizes
AES	128, 192, 256 bits
DES	64 bits including the parity bits. This gives an effective key size of 56 bits.
DES3	128 or 192 including the parity bits. This gives effective key sizes of 112 or 168 bits.
HMAC_MD5	Between 64 and 512 bits, multiple of 8 bits.
HMAC_SHA1	Between 80 and 512 bits, multiple of 8 bits
HMAC_SHA224	Between 112 and 512 bits, multiple of 8 bits.
HMAC_SHA256	Between 192 and 1024 bits, multiple of 8 bits
HMAC_SHA384	Between 256 and 1024 bits, multiple of 8 bits.
HMAC_SHA512	Between 256 and 1024 bits, multiple of 8 bits.
RSA_PUBLIC_KEY	Between 256 and 4096 bits, multiple of 64 bits.

RSA_KEYPAIR	
DSA_PUBLIC_KEY DSA_KEYPAIR	DSA_SHA1: 512 <= pbits <= 1024 (multiple of 64 bits), qbits = 160 DSA_SHA224: pbits = 2048, qbits = 224 DSA_SHA256: pbits = 3072, qbits = 256512 <= pbits <= 3072 and 160 <= qbits <= 256 (in steps of 8 bits), regardless of hash function.
DH_KEYPAIR	Between 256 and 2048 bits, multiple of 8 bits.
ECDSA_PUBLIC_KEY 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
ECDH_PUBLIC_KEY 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 4: Supported GP cryptographic algorithms and key sizes

Algorithm (TEE_ALG_*)	**Possible Modes**
AES_ECB_NOPAD, AES_CBC_NOPAD, AES_CTR, AES_CTS, AES_XTS, AES_CCM, AES_GCM	TEE_MODE_ENCRYPT TEE_MODE_DECRYPT
AES_CBC_MAC_NOPAD, AES_CBC_MAC_PKCS5, AES_CMAC	TEE_MODE_MAC
DES_ECB_NOPAD, DES_CBC_NOPAD, DES3_ECB_NOPAD, DES3_CBC_NOPAD	TEE_MODE_ENCRYPT TEE_MODE_DECRYPT
DES_CBC_MAC_NOPAD, DES_CBC_MAC_PKCS5 DES3_CBC_MAC_NOPAD, DES3_CBC_MAC_PKCS5	TEE_MODE_MAC
RSAES_PKCS1_V1_5 RSAES_PKCS1_OAEP_MGF1_SHA1 RSAES_PKCS1_OAEP_MGF1_SHA224 RSAES_PKCS1_OAEP_MGF1_SHA256 RSAES_PKCS1_OAEP_MGF1_SHA384 RSAES_PKCS1_OAEP_MGF1_SHA512 RSA_NOPAD	TEE_MODE_ENCRYPT TEE_MODE_DECRYPT
RSASSA_PKCS1_V1_5_MD5, RSASSA_PKCS1_V1_5_SHA1 RSASSA_PKCS1_V1_5_SHA224, RSASSA_PKCS1_V1_5_SHA256 RSASSA_PKCS1_V1_5_SHA384, RSASSA_PKCS1_V1_5_SHA512 RSASSA_PKCS1_PSS_MGF1_SHA1,	TEE_MODE_SIGN TEE_MODE_VERIFY

RSASSA_PKCS1_PSS_MGF1_SHA224 RSASSA_PKCS1_PSS_MGF1_SHA256 RSASSA_PKCS1_PSS_MGF1_SHA384 RSASSA_PKCS1_PSS_MGF1_SHA512 DSA_SHA1, DSA_SHA224, DSA_SHA256 ECDSA_SHA192, ECDSA_SHA224, ECDSA_SHA256, ECDSA_SHA384, ECDSA_SHA521	
DH, ECDH_SHA192, ECDH_SHA224, ECDH_SHA256, ECDH_SHA384, ECDH_SHA521	TEE_MODE_DERIVE
MD5, SHA1, SHA224, SHA256, SHA384, SHA512	TEE_MODE_DIGEST

Table 5: Supported GP algorithmic modes

4.2.1.6.2 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.

Legacy APIs supported key sizes and supported elliptic curves are equivalent to their GP counterpart, except for the DSA signature (DSA_RAW, DSA_HASHED) where key sizes can be: 512 <= pbits <= 3072 and 160 <= qbits <= 256 (in steps of 8 bits), regardless of hash function

Supported Cipher algorithms (TLAPI_ALG_*)
AES_128_CBC_NOPAD, AES_128_CBC_ISO9797_M1, AES_128_CBC_ISO9797_M2 AES_128_CBC_PKCS5, AES_128_CBC_PKCS7, AES_128_ECB_NOPAD, AES_128_CTR_NOPAD, AES_128_ECB_ISO9797_M1, AES_128_ECB_ISO9797_M2, AES_128_ECB_PKCS5, AES_128_ECB_PKCS7
AES_256_CBC_NOPAD, AES_256_CBC_ISO9797_M1, AES_256_CBC_ISO9797_M2, AES_256_CBC_PKCS5, AES_256_CBC_PKCS7, AES_256_ECB_NOPAD, AES_256_CTR_NOPAD, AES_256_ECB_ISO9797_M1, AES_256_ECB_ISO9797_M2, AES_256_ECB_PKCS5, AES_256_ECB_PKCS7
DES_CBC_ISO9797_M1, DES_CBC_ISO9797_M2 DES_CBC_NOPAD, DES_CBC_PKCS5 DES_ECB_ISO9797_M1, DES_ECB_ISO9797_M2 DES_ECB_NOPAD, DES_ECB_PKCS5
3DES_2KEY_CBC_ISO9797_M1, 3DES_2KEY_CBC_ISO9797_M2 3DES_2KEY_CBC_NOPAD, 3DES_2KEY_CBC_PKCS5 3DES_3KEY_CBC_ISO9797_M1, 3DES_3KEY_CBC_ISO9797_M2 3DES_3KEY_CBC_NOPAD, 3DES_3KEY_CBC_ PKCS5

RSA_ISO14888, RSA_NOPAD, RSA_PKCS1 RSA_SHA1_OAEP, RSA_SHA224_OAEP, RSA_SHA256_OAEP RSA_SHA384_OAEP, RSA_SHA512_OAEP RSACRT_SHA1_OAEP, RSACRT_SHA224_OAEP, RSACRT_SHA256_OAEP RSACRT_SHA384_OAEP, RSACRT_SHA512_OAEP Between 256 and 4096 bits, multiple of 64 bits.

Table 6 Supported Cipher Algorithms in Legacy API

Supported Digest Algorithms (TLAPI_ALG_*)
MD5, SHA1, SHA224, SHA256, SHA384, SHA512

Table 7 Supported Digest Algorithms in Legacy API

Supported Signature algorithms (TLAPI_SIG_*)
HMAC_MD5, HMAC_SHA1, HMAC_SHA224, HMAC_SHA_256, HMAC_SHA384, HMAC_SHA512
RSA_SHA_PKCS1, RSA_SHA224_PKCS1, RSA_SHA256_PKCS1, RSA_SHA384_PKCS1, RSA_SHA512_PKCS1, RSA_SHA1_PSS, RSA_SHA224_PSS, RSA_SHA256_PSS, RSA_SHA384_PSS, RSA_SHA512_PSS
DSA_RAW, DSA_HASHED
ECDSA_RAW, ECDSA_HASHED

Table 8 Supported Signature Algorithms in Legacy API

4.2.1.7 RNG

Kinibi provides a random number generator with sufficient entropy for cryptographic purposes. Seeded using the underlying platform RNG, Kinibi offers a DRBG corresponding to NIST SP 800-90A Hash-DRBG with SHA256 algorithm.

4.2.1.8 Device identification

Kinibi provides a service for retrieving the unique device identifier (provided by the underlying secure firmware).

4.2.1.9 Secure time

The TOE 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.

In the Time&RollbackProtection (T&R) mode: Persistent time is also enforced.

4.3 Scope of evaluation

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 TEE hardware architecture, identified by 3 classes of components:
  - Secure world dedicated hardware components
  - Normal world dedicated hardware components
  - Shared hardware components
  - (Includes MMU, TZPC, TZASC, GIC, …)
- The Secure Boot Software parts used to set up the TEE
- The platform TEE components (e.g. S-EL3 software as Secure Monitor / ATF)
- The Rich Execution Environment, including Kinibi Driver / Daemon and Client Applications

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

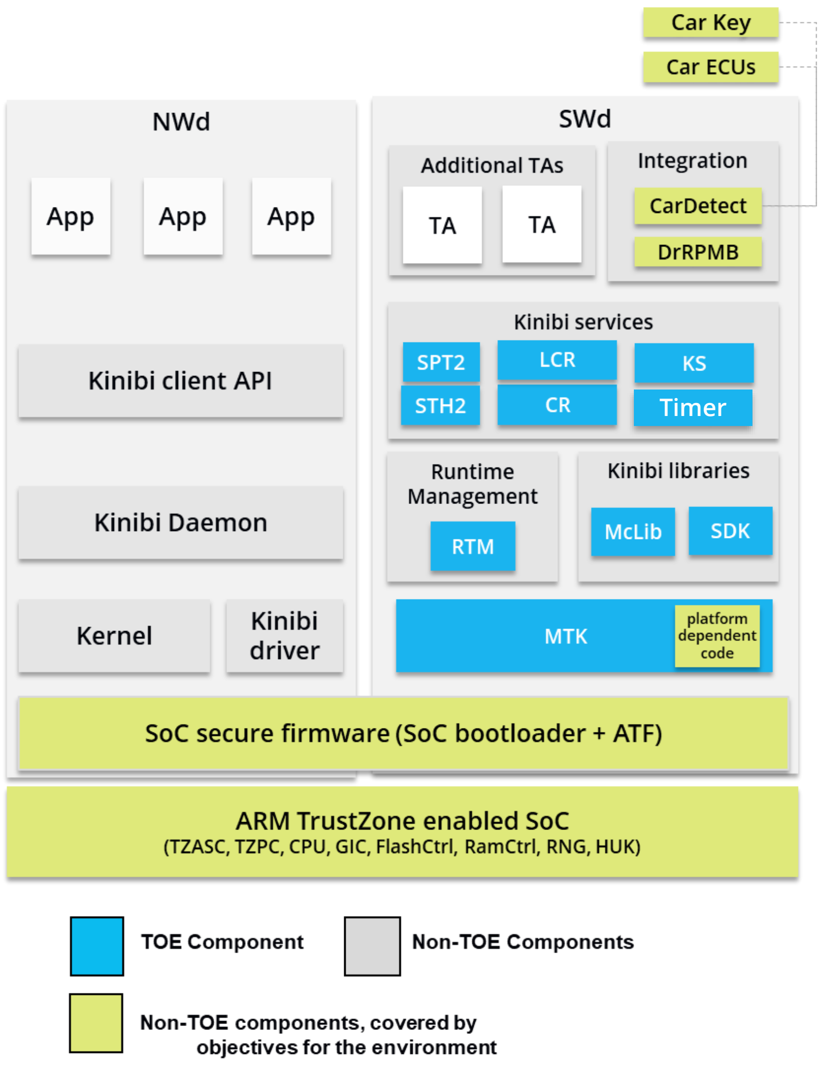

Figure 1: Illustration of Software components

TOE Software components	Descriptions
MTK	Microkernel running in the Secure world, which provides isolation between tasks, interprocess communication, and preemptive scheduling. In addition, the kernel hosts fastcall handlers for message exchange with the REE and architecture and porting layers to interface with the hardware. Note: Some parts Hardware specific parts of MTK must be adapted during the porting phase (before product delivery).
RTM	Task in Kinibi which handles memory management, session management, task loading, message passing and exception handling.
McLib	Code library that can be used by TAs and drivers in Kinibi. Includes the Kinibi proprietary Internal API and a subset of the GP TEE Internal API.
CR/LCR	Crypto drivers, providing cryptographic services to TAs and to other drivers. CR and LCR run as a separate task with driver privileges.
STH2	Storage driver, providing secure storage services to TAs and to other drivers. STH2 (STorage Handler v2) runs as a separate task with driver privileges.
SPT2	Secure Storage Proxy, a communication relay between the normal world and STH2. Abbreviated SPT2 (Storage Proxy Task for STH2).
Timer	Secure timer driver, providing secure timeout functionality for the KS.
KS	KillSwitch, keeps the TEE in a functional state until CarDetect TA stops resetting its timer.
SDK	Set of stubs for Trusted Application to invoke McLib APIs and Kinibi services.

Table 9 TOE Software components (part of Trusted OS image)

Non-TOE Hardware components	Description
TZASC	Firewall hardware to control all access to DRAM resource
TZPC	Peripheral controlling hardware to have TrustZone configurations for SoC
FlashCtrl	On-SoC flash controller
RamCtrl	On-SoC DRAM controller
CPU	Central Processing Unit

GIC	Interrupt controller
HUK	Hardware Unique Key (OTP Fuse)
RNG	Random Number Generator, which provides fresh entropy

Table 10 non-TOE Hardware components

Non-TOE Software components	**Description**
Bootloader	Component that initializes hardware for software execution, loads software image from flash storage to RAM for execution.
ATF – Secure monitor	Component that provides ARM TrustZone monitor functions including memory protection, interface for world switching, and loading other boot component.
DrRPMB	Secure driver, providing read/write interface for TA to store integrity data to RPMB partition (Embedded in the TEE image)
CarDetect	Component that keeps resetting the KillSwitch timer until it detects that the TEE hardware is in not in its legitimate environment.

Table 11 non-TOE Software components

External component	**Description**
Car key	Physical device required to start the car. It is required to set the TEE in a functional mode and to keep it in this state.
Car ECUs	Physical components of the car used to operate the vehicle. Some of them may be required to set the TEE in a functional mode and to keep it in this state.

Table 12 External component

4.4 TOE Environment: Required Hardware/Firmware/Software

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

- Cortex-A73 / Cortex-A53, ARMv8-A 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.
- Flash memory with RPMB support for storing TAs version and storage counters
- 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 (Secure Payload Dispatcher) patches applied. This patch ensures that REE SMC calls are forwarded to Kinibi OS.
- The platform-specific callbacks in the SPD patch must be implemented.

Table 12 specifies the minimum system requirements for the proper operation of the TOE when deployed on a hardware platform.

Category	**Requirement**
Processors	Cortex-A73, Cortex-A53, maximum 8 cores *(Such as Huawei Kirin 960)*
Flash Partition	Minimum 320 KB, recommended 512 KB With RPMB dedicated blocks (minimum 128KB) - (Not required for Hikey setup, software dummy key used instead).
Memory	Minimum 5 MB protected DDR
HW Identity	96 bits SoC ID unique among all chips of this silicon provider
HW Unique Key	256 bits SoC-specific key for the TEE (Not required for Hikey setup, software dummy key used instead).
RNG	64 bits of random data on each boot for the TEE

Table 13 TOE Minimum Requirements

4.5 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 TOE life cycle is split in five phases (Kinibi OS image delivery is done after phase 2):

- 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 SoC secure firmware, the Kinibi Trusted OS, any pre-installed Trusted Application and Secure 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 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 adapts the platform-specific parts of Kinibi to this chip and integrates the Kinibi related 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 OEM and the SIP develop, prepare and validate the software of the final product, including the Kinibi OS, required RPMB driver and additional Secure 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. Most secure services require an architecture that includes Trusted Applications and Secure Drivers. Today, common REEs require a TEE with a suitable suite of TA’s 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. When developing TAs, the respective developers shall follow the guidance [TR-DEVAPI]. When developing Secure Drivers, developers shall follow the guidance [TR-DRVAPI] to maintain the security objectives for the environment (covered in OE.TRUSTED_FIRMWARE).

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 Trusted Application and Secure Drivers 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 Trusted OS functionalities to the Trusted Applications and indirectly to the Client Applications within the REE..

In phase 5, SIP/OEM Trusted Application and Secure Drivers are loaded from the REE filesystem and installed in the TEE.

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

Note: In phase 5, SIP/OEM can sign new Secure Drivers and load them on the field through a REE image update, but this operation is considered as an update of the TEE code.

Table 14 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**
Phase 1: TEE OS Software design & Development	The TEE OS software developer is in charge of • TEE OS software development and testing • REE integration software development and testing
Phase 2: TEE OS software porting & delivery	The TEE OS software developer and the SIP SoC developer are in charge of: • Adapting the TEE OS to the SoC • The secure delivery of the TEE OS image and SDK to the SIP
Phase 3: SIP and OEM integration	The silicon vendor produces the chipset following the TEE OS requirements. The device manufacturer is responsible for the integration, validation and preparation of the software to load in the product that will include the Secure Firmware, the TEE OS, any Trusted Application pre-installed in the REE filesystem, 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 14 Actors in the Device Life Cycle

5 Security Problem Definition

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

5.1 Assets

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

TEE identification

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

Properties: unique and non-modifiable.

Application Note:

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

RNG

Random Number Generator.

Properties: unpredictable random numbers, sufficient entropy.

TA code

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

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

(T&R) Integrity of TA code means that the version successfully loaded it superior or equal to the last execution of this same TA identity.

TA data and keys

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

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

(T&R) Integrity of storage means that the value successfully read from a storage location is the last value that was written to this location.

TA instance time

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

Properties: monotonicity.

TEE runtime data

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

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

TEE persistent data

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

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

(T&R) Integrity of storage means that the value successfully read from a storage location is the last value that was written to this location.

TEE firmware

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

Properties: authenticity, integrity.

TEE initialization code and data

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

Properties: integrity.

TEE storage root of trust

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

Properties: integrity and confidentiality.

Application Note:

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

TA persistent time (T&R)

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

Properties: monotonicity.

TEE rollback detection data (T&R)

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

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 and Kinibi legacy APIs. Secure Drivers are a special class of Trusted Application that have enhanced privileges which enable use of the Kinibi Driver API's..

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 hardware or software means. Attackers are individuals or organizations with remote or physical (local) access to the automotive component embedding the TEE. Users with physical access to the car and the car key are not considered potential attackers.

The possible threats can be classified into the following categories:

Attacks through either wireless or standard physical connection of other equipment, to the vehicle systems.

Attacks from vehicle system software outside of the TEE.

The "threats" statement provides the general goal, the assets threatened and in some cases, typical 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.

T.IMPERSONATION

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

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

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

T.ROGUE_CODE_EXECUTION

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

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

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

Application Note:

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

T.PERTURBATION

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

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

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

Application Note:

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

T.RAM

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

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

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

Application Note:

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

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).

T.TEE_FIRMWARE_DOWNGRADE

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

Assets threatened directly (integrity): TEE firmware.

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

T.STORAGE_CORRUPTION

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

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

Application Note:

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

(T&R) The following two threats apply. Moreover, the standard threat T.STORAGE_CORRUPTION is no longer linked to OE.ROLLBACK but to O.ROLLBACK_PROTECTION.

T.ROLLBACK $^{(T\&R)}$

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

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

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

Application Note:

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

T.TA_PERSISTENT_TIME_ROLLBACK $^{(T\&R)}$

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

Assets threatened directly (integrity): TA persistent time.

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

Application Note:

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

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.INTEGRATION_CONFIGURATION

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

Application Note:

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

OSP.SECRETS

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

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.TA_MANAGEMENT

TA management software shall be developed 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.

The following assumptions hold on the TEE operational environment.

A.PROTECTION_AFTER_DELIVERY

It is assumed that the TOE is protected by the environment after delivery and before entering the final usage phase. It is assumed that the persons manipulating the TOE in the operational environment apply the **Trusted OS** guidelines **(Kinibi Integration Guide; Kinibi Driver Developers Guide)**. It is also assumed that the persons responsible for the application of the procedures contained in the guides, and the persons involved in delivery and protection of the product have the required skills and are aware of the security issues.

Application Note:

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

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.

Drivers are a special kind of TAs with additional privileges and APIs. Therefore, Driver developers are considered TA developers. Driver developers must follow all guidelines for TA development and additional guidelines for Driver development set by the TEE provider.

A.ROLLBACK

In TOE normal configuration mode, 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.CONFIGURATION

It is assumed that the TOE will be properly configured and installed 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. [Kinibi Integration Guide].

A.CONNECT

This assumption addresses 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.

Application note: Kinibi kernel addresses defined in the EL1 vector table is the only legitimate entry point to the TEE OS.

It is assumed that the SoC protects the communication of internal (TA and Kinibi OS) data that are transmitted in clear to physically separated peripherals to ensure the integrity and confidentiality of the data transmitted.

Application note: By design Kinibi protects the Flash stored contents (Trusted Storage) but not the RAM.

A.PEER.FUNC

It is assumed that the non-TOE hardware, firmware and software that are required for the TSF operation behave as expected.

Application note: This concerns for instance, secure boot function, CPU exception levels, memory management unit, TrustZone isolation integration, OTP memory and RPMB integration.

A.PEER.MGT

It is assumed that the non-TOE hardware, firmware and software that are required for the TSF operation are configured and managed to provide the expected security services to the TSF.

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 and the rollback protection 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 platform 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.CAR_PHYSICAL_PROTECTION

It is assumed that:

- the TEE operational environment isolates the TEE hardware from physical probing and manipulation.
- the TEE operational environment boots the TEE OS only when a connection with the physical car key is established.
- the TEE operational environment stops any TEE OS processing as soon as a failed connection with the physical car key is detected.

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

O.KEYS_USAGE

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

O.TEE_ID

The TOE shall provide means to retrieve the unique TOE 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 TOE 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 TOE shall check the validity of any operation requested from either the REE or a TA, at any entry point into the TOE
- 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 [TEE SA 1.2])

- Entry points (cf. [TEE SA 1.2]): Software in the REE must not be able to call directly to TEE Functions or Trusted OS 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 TOE runtime data and TA data and keys are protected against unauthorized disclosure. In particular,

- The TOE shall not export any sensitive data, random numbers or secret keys to the REE
- The TOE shall grant access to sensitive data, random numbers or secret keys only to authorized TAs
- The TOE 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 TOE OS, the TOE OS runtime data, the TA code and the TA data and keys are protected against unauthorized modification at runtime when stored in volatile memory.

O.TA_AUTHENTICITY

The TOE shall verify code authenticity of Trusted Applications.

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 TOE 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.

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

The TOE 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.ROLLBACK_PROTECTION(T&R)

The TOE shall prevent unauthorized rollback by:

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

Application Note:

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

O.TA_PERSISTENT_TIME(T&R)

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

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

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

OE.PROTECTION_AFTER_DELIVERY

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

Application Note:

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

OE.ROLLBACK

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

(T&R) this objective does not apply.

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.CONFIGURATION

It is assumed that the TOE will be properly configured and installed 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. (Kinibi Integration Guide).

OE.INITIALIZATION

It is assumed that the TOE is started through a secure initialization process starting from a non-modifiable boot code (ROM) that ensures:

o the integrity of the SoC secure firmware initialization code and data used to load the SoC secure firmware;

o the authenticity and rollback prevention of any secure boot stage required to initialize the TOE. (the SoC secure firmware includes all components of the secure boot chain)

o the authenticity and rollback prevention of the TOE image (including Kinibi trusted OS and embedded secure drivers);

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. [TEE SA 1.2]).

OE.TRUSTED_HARDWARE

SoC Hardware and secure Firmware implements 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 and are sufficiently protected from any attack[3] that may cause those functions to provide false results. In particular: An ARMv8 platform with REE/TEE isolation through TrustZone technology [ARM-TZ]. The trusted platform closes any debugging facilities. 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

The developers of SoC secure firmware and secure drivers are competent and trustworthy. They are capable and willing to ensure that the SoC secure firmware and the secure drivers does not break any

[3] In this ST, only attacks with moderate attack potential are considered.

security guarantee of the TOE, and they actually do so. Driver developers comply with the Kinibi Driver Developers Guide, in addition to the Kinibi Developers Guide which also hold for drivers.

OE.UNIQUE_TEE_ID

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

OE.RNG

The platform shall provide a random number generator (through the secure firmware HAL) 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. 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.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.CAR_PHYSICAL_PROTECTION

The hardware where the TEE is integrated shall be protected against physical probing and manipulation. This can be achieved through a physical layer such as an epoxy layer and a metal box containing hardware running the TOE (similar to a car Immobilizer ECU).
The CarKeyDetect TA is integrated in the TEE image. This TA establishes a secure connection with the car key and possibly other car component, and once established it turns the TEE in an operational mode and keeps it in this state until the connection is lost.

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 a proper management of failures
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data
- O.RUNTIME_INTEGRITY ensures protection against unauthorized modification of security functionality at runtime
- O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified
- O.TEE_DATA_PROTECTION ensures that the data used by the TEE are authentic and consistent
- O.TEE_ISOLATION enforces the separation between the TEE and the outside (REE and TAs)
- O.KEYS_USAGE controls the usage of cryptographic keys
- OE.TA_DEVELOPMENT enforces TA development principles, which are meant in particular to prevent disclosing information or performing modifications upon request of unauthorized entities.

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

- O.TEE_ID provides the unique TEE identification means
- OE.INITIALIZATION ensures that the TEE is bound to the SoC of the device
- O.RUNTIME_CONFIDENTIALITY prevents exposure of confidential data, in particular TSF data used to bind the TEE to the device
- O.RUNTIME_INTEGRITY prevents against unauthorized modification at runtime of security functionalities or data used to detect or prevent cloning
- O.TEE_DATA_PROTECTION prevents the TEE from using TEE data that is inconsistent or not authentic
- O.TRUSTED_STORAGE ensures that the trusted storage is bound to the device and prevents the TEE from using data that is inconsistent or not authentic
- 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:

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

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

- OE.INITIALIZATION ensures that the TEE security functionality is correctly initialized

- O.INSTANCE_TIME ensures the reliability of instance time stamps
- O.OPERATION ensures correct operation of the security functionality and a proper management of failures
- O.RUNTIME_CONFIDENTIALITY covers runtime TEE data which might influence the behavior of the TEE
- O.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.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).
- O.TA_PERSISTENT_TIME (T&R) ensures the reliability of persistent time stamps
- OE.CAR _PHYSICAL_PROTECTION ensures the physical protection of the hardware running the TEE while in its legitimate environment and protects against lab level attacks while moved from its legitimate environment.

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.CAR _PHYSICAL_PROTECTION ensures the physical protection of the RAM used by the TEE while in its legitimate environment and prevent any TEE processing while being moved out.

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

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

- OE.CAR _PHYSICAL_PROTECTION ensures the physical protection of the hardware running the TEE while in its legitimate environment and protects against lab level attacks while moved from its legitimate environment.

T.TEE_FIRMWARE_DOWNGRADE The combination of the following objectives ensures protection against TEE firmware downgrade:

- OE.INITIALIZATION ensures that the firmware that is executed is the version that was intended
- OE.INTEGRATION_CONFIGURATION ensures that the firmware installed in the device is the version that was intended
- OE.PROTECTION_AFTER_DELIVERY ensures that the firmware has not been modified after delivery.

T.STORAGE_CORRUPTION The combination of the following objectives ensures protection against corruption of non-volatile storage:

- O.OPERATION ensures the correct operation of the TEE security functionality, including storage
- O.TEE_DATA_PROTECTION ensures that stored TEE data are genuine and consistent
- O.TRUSTED_STORAGE enforces detection of corruption of the TA's storage
- O.TA_AUTHENTICITY ensures that the authenticity of TA code is verified
- OE.INITIALIZATION ensures that the firmware that is executed is the version that was intended
- OE.ROLLBACK states the limits of the properties enforced by the TSF.
- (T&R) The threat T.STORAGE_CORRUPTION is not linked to OE.ROLLBACK but to O.ROLLBACK_PROTECTION

T.ROLLBACK **(T&R)** The objective O.ROLLBACK_PROTECTION ensures the protection against rollback attacks.

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

6.3.2 Organizational Security Policies

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

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

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

6.3.3 Assumptions

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

A.ROLLBACK The objective OE.ROLLBACK directly covers this assumption. *Does not apply in (T&R*)

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

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 and OE.TRUSTED_FIRMWARE:

- 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 and OE.TRUSTED_FIRMWARE 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.CAR_PHYSICAL_PROTECTION OE.CAR_PHYSICAL_PROTECTION satisfies the assumption that the TEE is sufficiently protected against physical attacks.

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, O.KEYS_USAGE, OE.TA_DEVELOPMENT O.TA_AUTHENTICITY	Section 6.3.1

Threats	Security Objectives	Rationale
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 6.3.1
T.FLASH_DUMP	O.TRUSTED_STORAGE	Section 6.3.1
T.IMPERSONATION	O.CA_TA_IDENTIFICATION, O.OPERATION, O.RUNTIME_INTEGRITY, OE.TA_MANAGEMENT	Section 6.3.1
T.ROGUE_CODE_EXECUTION	OE.INITIALIZATION, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY, OE.TA_MANAGEMENT O.TA_AUTHENTICITY,	Section 6.3.1
T.PERTURBATION	OE.INITIALIZATION, O.INSTANCE_TIME, O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TA_PERSISTENT_TIME, O.TEE_DATA_PROTECTION, O.TEE_ISOLATION, O.TA_AUTHENTICITY, OE.CAR_PHYSICAL_PROTECTION	Section 6.3.1
T.RAM	OE.INITIALIZATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION, OE.CAR_PHYSICAL_PROTECTION	Section 6.3.1
T.RNG	OE.INITIALIZATION, O.RNG, OE.RNG, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY	Section 6.3.1
T.SPY	O.RUNTIME_CONFIDENTIALITY, O.TA_ISOLATION, O.TEE_ISOLATION, O.TRUSTED_STORAGE, OE.CAR_PHYSICAL_PROTECTION	Section 6.3.1
T.TEE_FIRMWARE_DOWNGRADE	OE.INITIALIZATION, OE.INTEGRATION_CONFIGURATION, OE.PROTECTION_AFTER_DELIVERY	Section 6.3.1
T.STORAGE_CORRUPTION	O.OPERATION, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE, O.TA_AUTHENTICITY, OE.INITIALIZATION, OE.ROLLBACK, O.ROLLBACK_PROTECTION,	Section 6.3.1

Threats	Security Objectives	Rationale
T.ROLLBACK	O.ROLLBACK_PROTECTION	Section 6.3.1
T.TA_PERSISTENT_TIME_ROLLBACK	O.TA_PERSISTENT_TIME	Section 6.3.1

Table 15 Threats and Security Objectives - Coverage

Security Objectives	Threats
O.CA_TA_IDENTIFICATION	T.IMPERSONATION
O.KEYS_USAGE	T.ABUSE_FUNCT
O.TEE_ID	T.CLONE
O.INSTANCE_TIME	T.PERTURBATION
O.OPERATION	T.ABUSE_FUNCT, T.IMPERSONATION, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.STORAGE_CORRUPTION
O.RNG	T.CLONE, T.RNG
O.RUNTIME_CONFIDENTIALITY	T.ABUSE_FUNCT, T.CLONE, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.RAM, T.RNG, T.SPY
O.RUNTIME_INTEGRITY	T.ABUSE_FUNCT, T.CLONE, T.IMPERSONATION, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.RAM, T.RNG
O.TA_AUTHENTICITY	T.ABUSE_FUNCT, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.STORAGE_CORRUPTION
O.TA_ISOLATION	T.PERTURBATION, T.RAM, T.SPY
O.TEE_DATA_PROTECTION	T.ABUSE_FUNCT, T.CLONE, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.STORAGE_CORRUPTION
O.TEE_ISOLATION	T.ABUSE_FUNCT, T.PERTURBATION, T.RAM, T.SPY
O.TRUSTED_STORAGE	T.CLONE, T.FLASH_DUMP, T.ROGUE_CODE_EXECUTION, T.SPY, T.STORAGE_CORRUPTION
O.ROLLBACK_PROTECTION	T.STORAGE_CORRUPTION, T.ROLLBACK
O.TA_PERSISTENT_TIME	T.PERTURBATION, T.TA_PERSISTENT_TIME_ROLLBACK
OE.INITIALIZATION	T.ABUSE_FUNCT, T.CLONE, T.ROGUE_CODE_EXECUTION, T.PERTURBATION, T.RAM, T.RNG, T.TEE_FIRMWARE_DOWNGRADE, T.STORAGE_CORRUPTION

Security Objectives	Threats
OE.INTEGRATION_CONFIGURATION	T.ROGUE_CODE_EXECUTION, T.TEE_FIRMWARE_DOWNGRADE
OE.PROTECTION_AFTER_DELIVERY	T.ROGUE_CODE_EXECUTION, T.TEE_FIRMWARE_DOWNGRADE
OE.ROLLBACK	T.STORAGE_CORRUPTION
OE.SECRETS	
OE.TA_DEVELOPMENT	T.ABUSE_FUNCT
OE.CONFIGURATION	
OE.UNIQUE_TEE_ID	
OE.TRUSTED_HARDWARE	
OE.TRUSTED_FIRMWARE	
OE.RNG	
OE.TA_MANAGEMENT	
OE.CAR_PHYSICAL_PROTECTION	T.PERTURBATION, T.RAM, T.SPY

Table 16 Security Objectives and Threats - Coverage

Organisational Security Policies	Security Objectives	Rationale
OSP.INTEGRATION_CONFIGURATION	OE.INTEGRATION_CONFIGURATION	Section 6.3.2
OSP.SECRETS	OE.SECRETS	Section 6.3.2
OSP.TEE_ID	O.TEE_ID, OE.UNIQUE_TEE_ID, O.RNG	Section 6.3.2
OSP.TA_MANAGEMENT	OE.TA_MANAGEMENT	Section 6.3.2

Table 17 OSPs and Security Objectives - Coverage

Security Objectives	Organisational Security Policies
O.CA_TA_IDENTIFICATION	
O.KEYS_USAGE	
O.TEE_ID	OSP.TEE_ID
O.INSTANCE_TIME	
O.OPERATION	
O.RNG	OSP.TEE_ID
O.RUNTIME_CONFIDENTIALITY	
O.RUNTIME_INTEGRITY	
O.TA_AUTHENTICITY	
O.TA_ISOLATION	
O.TEE_DATA_PROTECTION	
O.TEE_ISOLATION	
O.TRUSTED_STORAGE	

Security Objectives	Organisational Security Policies
O.ROLLBACK_PROTECTION	
O.TA_PERSISTENT_TIME	
OE.INTEGRATION_CONFIGURATION	OSP.INTEGRATION_CONFIGURATIO N
OE.PROTECTION_AFTER_DELIVERY	
OE.ROLLBACK	
OE.SECRETS	OSP.SECRETS
OE.TA_DEVELOPMENT	
OE.UNIQUE_TEE_ID	OSP.TEE_ID
OE.INITIALIZATION	
OE.CONFIGURATION	
OE.TRUSTED_HARDWARE	
OE.TRUSTED_FIRMWARE	
OE.RNG	
OE.TA_MANAGEMENT	OSP.TA_MANAGEMENT

Table 18 Security Objectives and OSPs - Coverage

Assumptions	Security Objectives for the Operational Environment	Rationale
A.PROTECTION_AFTER_DELIVERY	OE.PROTECTION_AFTER_DELIVERY	Section 6.3.3
A.ROLLBACK	OE.ROLLBACK	Section 6.3.3
A.TA_DEVELOPMENT	OE.TA_DEVELOPMENT	Section 6.3.3
A.CONFIGURATION	OE.CONFIGURATION	Section 6.3.3
A.CONNECT	OE.TRUSTED_HARDWARE OE.TRUSTED_FIRMWARE	Section 6.3.3
A.PEER.FUNC	OE.TRUSTED_HARDWARE OE.TRUSTED_FIRMWARE	Section 6.3.3
A.PEER.MGT	OE.TRUSTED_HARDWARE OE.TRUSTED_FIRMWARE	Section 6.3.3
A.POWER_UP	OE.INITIALIZATION	Section 6.3.3
A.RNG	OE.RNG	Section 6.3.3
A.CAR_PHYSICAL_PROTECTION	OE.CAR_PHYSICAL_PROTECTION	Section 6.3.3

Table 19 Assumptions and Security Objectives for the Operational Environment - Coverage

Security Objectives for the Operational Environment	Assumptions
OE.INTEGRATION_CONFIGURATION	
OE.PROTECTION_AFTER_DELIVERY	A.PROTECTION_AFTER_DELIVER Y
OE.ROLLBACK	A.ROLLBACK

Security Objectives for the Operational Environment	Assumptions
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
OE.CAR_PHYSICAL_PROTECTION	A.CAR_PHYSICAL_PROTECTION

Table 20 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

7.1.1.3 Extended Component FCS_RNG.1

7.1.1.4 Description

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

Hierarchical to: No other components.

Management: No management activities are foreseen.

Audit

No actions are defined to be auditable.

7.1.1.5 Definition

FCS_RNG.1 Random numbers generation

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

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

Dependencies: No dependencies.

8 SECURITY FUNCTIONAL REQUIREMENTS

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
- 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
- FPT_TEE.1 Testing of external entities

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

- FCS_RNG.1 Random numbers generation

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

Users stand for entities outside the TOE:

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

Subjects stand for active entities insside the TOE:

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

- S.RESOURCE: any software or hardware component which may be used alternatively by the TEE or the REE, with security attribute "state" (TEE/REE). E.g. cryptographic accelerator, random number generator, cache, registers.
- S.RAM_UNIT: RAM addressable unit, with security attribute "rights:(TA identifier/REE) -> (Read/Write/ReadWrite/NoAccess). For instance, an addressable unit may be allocated or have its access rights changed upon TA instance creation or when sharing memory references between a client (CA, TA) and a TA. Notes: 1) A RAM_UNIT typically stands for a byte in the C language; 2) there is no RAM access restriction applicable to the TEE itself
- S.COMM_AGENT: proxy between CAs in the REE and the TEE and its TAs.

Objects stand for passive entities inside the TOE:

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

Cryptographic objects are a special kind of TEE objects:

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

Information stands for data exchanged between subjects:

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

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

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

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

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

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

Other operations:

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

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

Runtime Data Information Flow Control SFP:

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

TA Keys Access Control SFP:

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

Trusted Storage Access Control SFP:

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

8.1.1 Identification

FIA_ATD.1 User attribute definition

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

Application Note:

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

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

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

FIA_UID.2 User identification before any action

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

Application Note:

User stands for Client Application or Trusted Application.

FIA_USB.1 User-subject binding

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

- o **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:

- o **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**

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:

- o **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

- o **TSF**
- o **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 and SDs.

8.1.2 Confidentiality, Integrity and Isolation

FDP_IFC.2/Runtime Complete information flow control

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

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

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

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

Application Note:

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

TOE environment: Underlying Hardware and Secure Firmware must provide correct MMU operation and correct DRAM isolation configuration. (Covered in A.PEER.FUNC and A.PEER.MGT)

FDP_IFF.1/Runtime Simple security attributes

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

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

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

FDP_IFF.1.3/Runtime The TSF shall enforce **no additional information flow control SFP rule**.

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

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

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

Application Note:

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

TOE environment: Underlying Hardware, Secure Firmware and additional Secure Drivers must provide correct MMU operation and correct DRAM isolation configuration. (Covered by OE.TRUSTED_HARDWARE and OE.TRUSTED_FIRMWARE)

FDP_RIP.1/Runtime Subset residual information protection

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

Application Note:

This operation applies in particular upon:

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

FPT_ITT.1/Runtime Basic internal TSF data transfer protection

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

Refinement:

Separate part of the TOE stands for separate software components of the Trusted OS (micro-kernel, runtime manager, Secure Drivers, Trusted Applications)

8.1.3 Cryptography

FCS_COP.1 Cryptographic operation

FCS_COP.1.1 The TSF shall perform **cryptographic operations defined in table below** in accordance with a specified cryptographic algorithm **defined in table below** and **cryptographic key sizes defined in table below** that meet the following: **standards defined in table below:**

Authenticity of the Secure Drivers and TA code verification	*RSA_SHA256_PSS with key size > 2048* *HMAC-SHA256 with key size > 256* *AES-128 CBC* *SHA-256 truncated to 128 bits*
Consistency and confidentiality of Trusted Storage data	*HMAC-SHA256 with key size > 256* *AES-128 CBC* *SHA-256 truncated to 128 bits*
Cryptographic algorithms in legacy and GP TA apis.	*See tables in 4.2.1.6.1* GlobalPlatform API *and 4.2.1.6.2* Legacy API

Note: Some of the cryptographic algorithms specified by GlobalPlatform and made available to TAs through the GlobalPlatform APIs and Kinibi legacy APIs are no longer considered secure and are only provided for compatibility and shall not be used for security. The deprecated algorithms list includes:

- DES and 3DES symmetric algorithms
- MD5 and SHA1 hashing algorithms and all cryptographic algorithms based on them
- RSA operation with keys shorter than 2048 bits, and RSA key generation for any bit size
- DH operation with keys having a group shorter than 2048.

FDP_ACC.1/TA_keys Subset access control

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

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

FDP_ACF.1/TA_keys Security attribute based access control

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

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

- o **OP.USE_KEY is allowed if the following conditions hold:**
  - ▪ **The TA instance that requested the operation to the API owns the key (S.API.caller = OB.TA_KEY.owner)**
  - ▪ **The intended usage of the key (OB.TA_KEY.usage) matches the requested operation**

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

FDP_ACF.1.3/TA_keys The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **no additional rules**.

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

- o **Any access to a user key attempted directly from S.TA_INSTANCE or any other subject of the TEE that is not S.API**
- o **Any access to a user key attempted from S.API without valid caller (S.API.caller is undefined)**
- o **none**

Application Note:

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

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

FMT_MSA.1/TA_keys Management of security attributes

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

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

FMT_MSA.3/TA_keys Static attribute initialisation

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

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

8.1.4 Initialization, Operation and Firmware Integrity

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 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 caller. There is no recovery mechanism.**
- detection of TEE firmware integrity violation: **TSF will enter a halt state if it detects an impossible system state. In the halt state, TSF does not execute any TA code.**

TOE environment: Hardware is responsible for aborts on invalid memory accesses. (OE.TRUSTED_HARDWARE)

FDP_SDI.2 Stored data integrity monitoring and action

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

Refinement:

The TSF shall monitor TEE runtime data, TEE persistent data, TA data and keys and TA code stored in containers controlled by the TSF for **authenticity and consistency errors** on all objects, based on the following attributes: **signatures of TEE persistent data, signatures and checksums on TA data and keys, and signatures and consistency checks of 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 authenticity or consistency errors in TEE runtime data or TEE persistent data, the TSF shall **behave in a manner that does not depend on the compromised data**
- Upon detection of TA code 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**

Application Note:

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

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

FPT_FLS.1 Failure with preservation of secure state

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

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

Application Note:

- Device binding failure occurs when (part of) the stored data has not been bound by the same TEE
- Unrecoverable TA operation failure leads generally to TA being killed by the TEE OS.
- Unrecoverable TEE OS operation failures leads to TEE going silent, refusing any new operation from REE or any TA.

TOE Environment: Platform Hardware and Secure Firmware is assumed to respond securely to operation failures such as Platform and Secure Firmware initialization, TEE OS integrity or downgrade check failure, invalid command handling in the Secure Firmware and Hardware operation failure (OE.INITIALIZATION, OE.TRUSTED_HARDWARE, OE.TRUSTED_SOFTWARE)

FMT_SMF.1 Specification of Management Functions

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

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

FPT_TEE.1 Testing of external entities

FPT_TEE.1.1 The TSF shall run a suite of tests **prior execution and 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**.

8.1.5 TEE Identification

FAU_SAR.1 Audit review

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

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

8.1.6 Instance Time

FPT_STM.1/Instance time Reliable time stamps

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

Refinement:

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

Application Note:

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

8.1.7 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). Kinibi DRBG is functionally compliant to NIST SP 800-90A but does not perform reseeding or health checks.

TOE Environment: Hardware RNG support is covered by OE.RNG.

8.1.8 Trusted Storage

FDP_ACC.1/Trusted Storage Subset access control

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

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

TOE Environment: Kinibi Trusted storage relies on a device specific secret key. Provisioning of this secret is covered by OE.SECRETS. Access to this secret is platform specific and integrated in the TEE OS during the phase 2 of the TOE lifecycle: "Kinibi porting for a specific Silicon Provider SoC".

FDP_ACF.1/Trusted Storage Security attribute based access control

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

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

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

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

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

- **Any access to a trusted storage attempted from S.API without valid caller (S.API.caller = undefined)**
- **Any access to a trusted storage that was bound to a different TEE (OB.TA_STORAGE.TEE_identity different from OB.SRT.TEE_identity)**
- **Any access to a trusted storage from a subject different from S.API**
- **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 unauthorized 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 **a failure occurring during any operation used to store persistent object (data and keys).**

Application Note:

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

FMT_MSA.1/Trusted Storage Management of security attributes

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

FMT_MSA.3/Trusted Storage Static attribute initialisation

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

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

8.1.9 Time&RollbackProtection (T&R) TOE configuration

This section only applies when the Rollback protection of Kinibi Secure Storage is activated.

The following security functional components defined in CC Part 2 [CC2] are used in this Security Target:

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

8.1.9.1 Rollback Protection

FDP_SDI.2/Rollback Stored data integrity monitoring and action

FDP_SDI.2.1/Rollback The TSF shall monitor user data stored in containers controlled by the TSF for **integrity errors** on all objects, based on the following attributes: **signatures and versions of TEE**

persistent data, signatures, checksums and versions of TA data and keys, and signatures, consistency checks and versions of TA code.

Refinement:

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

FDP_SDI.2.2/Rollback 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 integrity errors in TEE rollback detection data, TEE runtime data or TEE persistent data, the TSF shall **behave in a manner that does not depend on the compromised data**
- Upon detection of TA code integrity errors, the TSF shall **abort the execution of the TA instance**
- Upon detection of TA data or TA keys integrity errors, the TSF shall
  - **Not provide any compromised data,**
  - **Behave in a manner that does not depend on the compromised data**

Application Note:

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

TOE environment: Kinibi leverages RPMB flash storage to enforce rollback protection on the Trusted Storage. RPMB service is offered to the Kinibi Trusted OS through a secure driver developed by the SIP with the help of Trustonic during the phase 3 of the TOE lifecycle: "SIP and OEM integration".

FPT_FLS.1/Rollback Failure with preservation of secure state

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

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

Application Note:

This requirement is a complement to FPT_FLS.1.

8.1.9.2 TA Persistent Time

FPT_STM.1/Persistent Time Reliable time stamps

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

Refinement:

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

- Time stamps are persistent over TEE reset

- o Time stamps are monotonic between two 'time setting' operations performed by any instance of the TA.

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

Application Note:

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

FMT_MTD.1/Persistent Time Management of TSF data

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

Application Note:

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

FMT_SMF.1/Persistent Time Specification of Management Functions

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

Application Note:

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

8.2 Security Requirements Rationale

8.2.1 Security Objectives for the TOE

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

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

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

- o FCS_COP.1 allows to specify the cryptographic operations in the scope of the evaluation if any
- o FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys, FMT_SMR.1 and FMT_SMF.1 state the key access policy, which grants access to the owner of the key only.

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

- o FAU_SAR.1 enforces TEE identifier access capabilities

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
- FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys and FMT_SMF.1 state the key access policy.

Rationale specific to the Time and Rollback PP-Module:

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

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

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

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

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

- FDP_SDI.2 enforces the consistency and authenticity of TA code
- FPT_TEE.1 enforces the check of authenticity of TA code prior execution
- FCS_COP.1 states the cryptography used to verify the authenticity of TA code

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

- FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage and FMT_SMF.1 state the policy for controlling access to TA storage

- FCS_COP.1 state the cryptographic algorithms used for Trusted Storage to ensure confidentiality and authenticity of TA data
- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state the policy for controlling access to TA execution space
- FPT_FLS.1 enforces TA isolation by maintaining a secure state, in particular in case of panic states

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

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

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

- FDP_IFC.2/Runtime and FDP_IFF.1/Runtime state the policy for controlling access to TEE execution space.

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

- FCS_COP.1 states the cryptography used to protect integrity and confidentiality of the TA data in external memory, if applicable
- FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage and FMT_SMF.1 state Storage state the policy for accessing TA trusted storage and protecting the confidentiality of data
- FDP_SDI.2 enforces the consistency and authenticity of the trusted storage
- FPT_FLS.1 maintains a secure state.

O.ROLLBACK_PROTECTION$^{\mathbf{(T\&R)}}$ The following requirements contribute to fulfill the objective:

- FDP_SDI.2/Rollback$^{(T\&R)}$ states the behavior of the TEE upon integrity failure (thus rollback)
- FPT_FLS.1/Rollback$^{(T\&R)}$ enforces the detection of integrity failure (thus rollback detection).

O.TA_PERSISTENT_TIME$^{\mathbf{(T\&R)}}$ The following requirements fulfill the objective:

- FPT_STM.1/Persistent Time$^{(T\&R)}$ states the persistent time reliability conditions expected from the TEE
- FMT_MTD.1/Persistent Time$^{(T\&R)}$ states the roles that can perform 'time-setting' operations
- FMT_SMF.1/Persistent Time$^{(T\&R)}$ states the existence of a 'time-setting' managemenent function.

8.2.2 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 8.3.1
O.KEYS_USAGE	FCS_COP.1, FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys, FMT_SMR.1, FMT_SMF.1	Section 8.3.1
O.TEE_ID	FAU_SAR.1	Section 8.3.1

Security Objectives	Security Functional Requirements	Rationale
O.INSTANCE_TIME	FPT_STM.1/Instance time	Section 8.3.1
O.OPERATION	FAU_ARP.1, FDP_SDI.2, FIA_ATD.1, FIA_UID.2, FIA_USB.1, FMT_SMR.1, FPT_FLS.1, FMT_SMF.1, FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FDP_ACC.1/TA_keys, FDP_ACF.1/TA_keys, FMT_MSA.1/TA_keys, FMT_MSA.3/TA_keys, FDP_SDI.2/Rollback(T&R), FPT_FLS.1/Rollback(T&R),	Section 8.3.1
O.RNG	FCS_RNG.1	Section 8.3.1
O.RUNTIME_CONFIDENTIALITY	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FPT_ITT.1/Runtime, FDP_RIP.1/Runtime	Section 8.3.1
O.RUNTIME_INTEGRITY	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FPT_ITT.1/Runtime, FDP_SDI.2	Section 8.3.1
O.TA_AUTHENTICITY	FDP_SDI.2, FCS_COP.1, FPT_TEE.1	Section 8.3.1
O.TA_ISOLATION	FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FMT_SMF.1, FCS_COP.1, FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FPT_FLS.1	Section 8.3.1
O.TEE_DATA_PROTECTION	FCS_COP.1, FDP_SDI.2, FPT_ITT.1/Runtime	Section 8.3.1
O.TEE_ISOLATION	FDP_IFC.2/Runtime, FDP_IFF.1/Runtime	Section 8.3.1
O.TRUSTED_STORAGE	FCS_COP.1, FDP_ACC.1/Trusted Storage, FDP_ACF.1/Trusted Storage, FDP_ROL.1/Trusted Storage, FMT_MSA.1/Trusted Storage, FMT_MSA.3/Trusted Storage, FMT_SMF.1, FDP_SDI.2, FPT_FLS.1	Section 8.3.1
O.ROLLBACK_PROTECTION(T&R)	FDP_SDI.2/Rollback(T&R), FPT_FLS.1/Rollback(T&R)	Section 8.3.1

Security Objectives	Security Functional Requirements	Rationale
O.TA_PERSISTENT_TIME(T&R)	FPT_STM.1/Persistent Time(T&R), FMT_MTD.1/Persistent Time(T&R), FMT_SMF.1/Persistent Time(T&R)	Section 8.3.1

Table 21 Security Objectives and SFRs - Coverage

Security Functional Requirements	Security Objectives
FIA_ATD.1	O.CA_TA_IDENTIFICATION, O.OPERATION
FIA_UID.2	O.CA_TA_IDENTIFICATION, O.OPERATION
FIA_USB.1	O.CA_TA_IDENTIFICATION, O.OPERATION
FMT_SMR.1	O.KEYS_USAGE, O.OPERATION
FDP_IFC.2/Runtime	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION
FDP_IFF.1/Runtime	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TA_ISOLATION, O.TEE_ISOLATION
FDP_RIP.1/Runtime	O.RUNTIME_CONFIDENTIALITY
FPT_ITT.1/Runtime	O.RUNTIME_CONFIDENTIALITY, O.RUNTIME_INTEGRITY, O.TEE_DATA_PROTECTION
FCS_COP.1	O.KEYS_USAGE, O.TA_AUTHENTICITY, O.TA_ISOLATION O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE
FDP_ACC.1/TA_keys	O.KEYS_USAGE, O.OPERATION
FDP_ACF.1/TA_keys	O.KEYS_USAGE, O.OPERATION
FMT_MSA.1/TA_keys	O.KEYS_USAGE, O.OPERATION
FMT_MSA.3/TA_keys	O.KEYS_USAGE, O.OPERATION
FAU_ARP.1	O.OPERATION
FDP_SDI.2	O.OPERATION, O.RUNTIME_CONFIDENTIALITY, O.TA_AUTHENTICITY, O.TEE_DATA_PROTECTION, O.TRUSTED_STORAGE,
FPT_FLS.1	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FMT_SMF.1	O.KEYS_USAGE, O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FPT_TEE.1	O.TA_AUTHENTICITY
FAU_SAR.1	O.TEE_ID
FPT_STM.1/Instance time	O.INSTANCE_TIME
FCS_RNG.1	O.TEE_ID, O.RNG
FDP_ACC.1/Trusted Storage	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FDP_ACF.1/Trusted Storage	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FDP_ROL.1/Trusted Storage	O.TRUSTED_STORAGE

Security Functional Requirements	Security Objectives
FMT_MSA.1/Trusted Storage	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FMT_MSA.3/Trusted Storage	O.OPERATION, O.TA_ISOLATION, O.TRUSTED_STORAGE
FDP_SDI.2/Rollback (T&R)	O.OPERATION, O.ROLLBACK_PROTECTION(T&R)
FPT_FLS.1/Rollback (T&R)	O.OPERATION, O.ROLLBACK_PROTECTION(T&R)
FPT_STM.1/Persistent Time(T&R)	O.TA_PERSISTENT_TIME(T&R)
FMT_MTD.1/Persistent Time(T&R)	O.TA_PERSISTENT_TIME(T&R)
FMT_SMF.1/Persistent Time(T&R)	O.TA_PERSISTENT_TIME(T&R)

Table 22 SFRs and Security Objectives

8.2.3 Dependencies

8.2.3.1 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)	
FMT_SMF.1	No Dependencies	
FPT_TEE.1	No Dependencies	
FDP_RIP.1/Runtime	No Dependencies	
FDP_IFC.2/Runtime	(FDP_IFF.1)	FDP_IFF.1/Runtime
FDP_IFF.1/Runtime	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/Runtime
FPT_ITT.1/Runtime	No Dependencies	
FCS_COP.1	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	
FDP_ACC.1/TA_keys	(FDP_ACF.1)	FDP_ACF.1/TA_keys
FDP_ACF.1/TA_keys	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/TA_keys, FMT_MSA.3/TA_keys
FMT_MSA.1/TA_keys	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FDP_ACC.1/TA_keys, FMT_SMF.1
FMT_MSA.3/TA_keys	(FMT_MSA.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_MSA.1/TA_keys
FAU_ARP.1	(FAU_SAA.1)	

Requirements	CC Dependencies	Satisfied Dependencies
FDP_SDI.2	No Dependencies	
FPT_FLS.1	No Dependencies	
FPT_STM.1/Instance time	No Dependencies	
FCS_RNG.1	No Dependencies	
FDP_ACC.1/Trusted Storage	(FDP_ACF.1)	FDP_ACF.1/Trusted Storage
FDP_ACF.1/Trusted Storage	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/Trusted Storage, FMT_MSA.3/Trusted Storage
FDP_ROL.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.1/Trusted Storage
FMT_MSA.1/Trusted Storage	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_SMF.1, FDP_ACC.1/Trusted Storage,
FMT_MSA.3/Trusted Storage	(FMT_MSA.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_MSA.1/Trusted Storage
FDP_SDI.2/Rollback$^{(T\&R)}$	No Dependencies	
FPT_FLS.1/Rollback$^{(T\&R)}$	No Dependencies	
FPT_STM.1/Persistent Time$^{(T\&R)}$	No Dependencies	
FMT_MTD.1/Persistent Time$^{(T\&R)}$	(FMT_SMF.1) and (FMT_SMR.1)	FMT_SMR.1, FMT_SMF.1/Persistent Time
FMT_SMF.1/Persistent Time$^{(T\&R)}$	No Dependencies	

Table 23 SFRs Dependencies

8.2.3.1.1 Rationale for the exclusion of Dependencies

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

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

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

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

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

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

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

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

8.2.3.2 SARs Dependencies

Requirements	CC Dependencies	Satisfied Dependencies
ADV_ARC.1	(ADV_FSP.1) and (ADV_TDS.1)	ADV_FSP.5, ADV_TDS.4
ADV_FSP.5	(ADV_TDS.1) and (ADV_IMP.1)	ADV_TDS.4, ADV_IMP.1
ADV_IMP.1	(ALC_TDS.3) and (ALC_TAT.1)	ALC_TDS.4, ALC_TAT.2
ADV_INT.2	(ADV_IMP.1) and (ADV_TDS.3) and (ALC_TAT.1)	ADV_IMP.1, ADV_TDS.4, ALC_TAT.2
ADV_TDS.4	(ADV_FSP.5)	ADV_FSP.5
AGD_OPE.1	(ADV_FSP.1)	ADV_FSP.5
AGD_PRE.1	No Dependencies	
ALC_CMC.4	(ALC_CMS.1) and (ALC_DVS.1) and (ALC_LCD.1)	ALC_CMS.5, ALC_DVS.1, ALC_LCD.1
ALC_CMS.5	No Dependencies	
ALC_DEL.1	No Dependencies	
ALC_DVS.1	No Dependencies	
ALC_FLR.1	No Dependencies	
ALC_LCD.1	No Dependencies	
ALC_TAT.2	(ADV_IMP.1)	ALC_IMP.1
ASE_CCL.1	(ASE_ECD.1) and (ASE_INT.1) and (ASE_REQ.1)	ASE_ECD.1, ASE_INT.1, ASE_REQ.2
ASE_ECD.1	No Dependencies	
ASE_INT.1	No Dependencies	
ASE_OBJ.2	(ASE_SPD.1)	ASE_SPD.1
ASE_REQ.2	(ASE_ECD.1) and (ASE_OBJ.2)	ASE_ECD.1, ASE_OBJ.2
ASE_SPD.1	No Dependencies	
ASE_TSS.1	(ADV_FSP.1) and (ASE_INT.1) and (ASE_REQ.1)	ADV_FSP.5, ASE_INT.1, ASE_REQ.2
ATE_COV.2	(ADV_FSP.2) and (ATE_FUN.1)	ADV_FSP.5, ATE_FUN.1

Requirements	CC Dependencies	Satisfied Dependencies
ATE_DPT.3	(ADV_ARC.1) and (ADV_TDS.4) and (ATE_FUN.1)	ADV_ARC.1, ADV_TDS.4, ATE_FUN.1
ATE_FUN.1	(ATE_COV.1)	ATE_COV.2
ATE_IND.2	(ADV_FSP.2) and (AGD_OPE.1) and (AGD_PRE.1) and (ATE_COV.1) and (ATE_FUN.1)	ADV_FSP.5, AGD_OPE.1, AGD_PRE.1, ATE_COV.2, ATE_FUN.1
AVA_VAN.4	(ADV_ARC.1) and (ADV_FSP.4) and (ADV_TDS.3) and (AGD_OPE.1) and (AGD_PRE.1)	ADV_ARC.1, ADV_FSP.5, ADV_TDS.4, AGD_OPE.1, AGD_PRE.1

Table 24 SARs Dependencies

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.

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

Kinibi locks any crypto processing using the device unique key and prevents any TA loading until the CarKeyConnect TA securely connects to the car key and other car components. As soon as the connection is lost, the CarKeyConnect TA stops triggering the DrKillSwitch which results in a TEE halt. These mechanisms prevent attackers from extracting the TOE from its legitimate environment so as to run physical attacks with lab equipment to extract cryptographic keys.

TA_CA_Identification

All TAs or CAs are assigned a unique identifier.

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

Trusted Application Downgrade protection (T&R)

The TOE verifies all TA version prior to execution to prevent downgrading to a previous version,

Runtime Data Information Flow Control

Isolated Execution: ensures TAs and internal TOE software components execute completely isolated from and unhindered by others and guarantees that any code and data is protected at runtime by leveraging the MMU Hardware component.

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.

(T&R): the TOE protects persistently stored data belonging to the TEE or TAs against rollback with Flash RPMB support.

Secure time

The TOE provides a service to retrieve a monotonic timestamp during a TA session lifetime. In **(T&R)** TOE configuration, the TOE provides a service to store and retrieve time stamps that are persistent over TEE reset and monotonic between two 'time setting' operations performed by any instance of the TA by storing it in the rollback protected Secure Filesystem. Isolation of the Hardware timestamp provider is covered by OE.TRUSTED_HARDWARE.

9.2 SFRs and TSS

9.2.1 SFRs and TSS - Rationale

9.2.1.1.1 Identification

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

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

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

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

9.2.1.1.2 Confidentiality, Integrity and Isolation

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.

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.

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

FDP_ACC.1/TA_keys This SFR is fully covered by Cryptographic Support covering cryptographic operation.

FDP_ACF.1/TA_keys This SFR is fully covered by Cryptographic Support covering cryptographic operation.

FMT_MSA.1/TA_keys This SFR is fully covered by Cryptographic Support covering cryptographic operation.

FMT_MSA.3/TA_keys This SFR is fully covered by Cryptographic Support covering cryptographic operation.

9.2.1.1.4 Initialization, Operation and Firmware Integrity

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

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_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.

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

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

9.2.1.1.5 TEE Identification

FAU_SAR.1 This SFR is fully covered by Device Identification covering TEE identifier access.

9.2.1.1.6 Instance Time

FPT_STM.1 This SFR is fully covered by Secure Time covering reliable timestamp access.

9.2.1.1.7 Random Number Generator

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

9.2.1.1.8 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 SFRs and TSS – Rationale (T&R)

FDP_SDI.2 This SFR is fully covered by Trusted Storage covering the management of Trusted Storage Access Control and Trusted Application Downgrade protection

FPT_FLS.1 This SFR is fully covered by Trusted Storage covering the management of Trusted Storage Access Control and Trusted Application Downgrade protection.

FPT_STM.1 This SFR is fully covered by Secure Time covering reliable timestamp access and by Trusted Storage covering the management of Trusted Storage Access Control.

FMT_MTD.1 This SFR is fully covered by Secure Time covering reliable timestamp access and by Trusted Storage covering the management of Trusted Storage Access Control.

FMT_SMF.1 This SFR is fully covered by Secure Time covering reliable timestamp access and by Trusted Storage covering the management of Trusted Storage Access Control.

9.2.2 Association tables of SFRs and TSS

Security Functional Requirements	TOE Summary Specification
FIA_ATD.1	TA_CA_Identification
FIA_UID.2	TA_CA_Identification
FIA_USB.1	TA_CA_Identification
FMT_SMR.1	TA_CA_Identification
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
FDP_RIP.1/Runtime	Runtime Data Information Flow Control
FCS_COP.1	Cryptographic Support
FDP_ACC.1/TA_Keys	Runtime Data Information Flow Control
FDP_ACF.1/TA_Keys	Runtime Data Information Flow Control
FMT_MSA.1/TA_Keys	Runtime Data Information Flow Control
FMT_MSA.3/TA_Keys	Runtime Data Information Flow Control
FAU_ARP.1	Operation and Firmware Integrity
FDP_SDI.2	Operation and Firmware Integrity
FPT_FLS.1	Operation and Firmware Integrity, Secure Initialization
FMT_SMF.1	Cryptographic Support Trusted Storage
FPT_TEE.1	Trusted Application Authentication
FAU_SAR.1	Device Identification
FPT_STM.1/Instance time	Secure time
FCS_RNG.1	Cryptographic Support
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
FDP_SDI.2/Rollback	Trusted Storage Trusted Application Downgrade protection
FPT_FLS.1/Rollback	Trusted Storage Trusted Application Downgrade protection
FPT_STM.1/Persistent Time	Trusted Storage

Security Functional Requirements	TOE Summary Specification
FMT_MTD.1/Persistent Time	Trusted Storage
FMT_SMF.1/Persistent Time	Trusted Storage

Table 25 SFRs and TSS - Coverage

TOE Summary Specification	Security Functional Requirements
TA_CA_Identification	FIA_UID.2, FMT_SMR.1, FIA_ATD.1, FIA_USB.1
Runtime Data Information Flow Control	FDP_RIP.1/Runtime, FDP_IFC.2/Runtime, FDP_IFF.1/Runtime, FPT_ITT.1/Runtime, FDP_ACC.1/TA_Keys, FDP_ACF.1/TA_Keys, FMT_MSA.1/TA_Keys, FMT_MSA.3/TA_Keys
Cryptographic Support	FCS_COP.1, FCS_RNG.1
Device Identification	FAU_SAR.1
Operation and Firmware Integrity	FDP_SDI.2, FAU_ARP.1, FPT_FLS.1
Secure Initialization	FPT_FLS.1
Trusted Application Authentication	FPT_TEE.1
Secure Time	FPT_STM.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, FDP_SDI.2/Rollback, FPT_FLS.1/Rollback, FPT_STM.1/Persistent Time, FMT_MTD.1/Persistent Time, FMT_SMF.1/Persistent Time
Trusted Application Downgrade protection	FDP_SDI.2/Rollback FPT_FLS.1/Rollback

Table 26 TSS and SFRs - Coverage

10 TECHNICAL TERMS, ABBREVIATION AND ASSOCIATED REFERENCES

10.1 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 stored data used at startup is the data for which runtime consistency was enforced on the same TEE on a previous power cycle. Consistency implies runtime integrity of what is successfully written and read back – values or code. However the stored data used at startup may be restored from an old power cycle, not the latest one. It is still consistent at start-up because it corresponds to a memory snapshot at a given time, but it represents an integrity loss compared with the latest power cycle. This notion is weaker than integrity that must be preserved between power cycles.
Device binding	Device binding is the property of data being only usable on a unique given system instance, here a TEE.
Execution Environment (EE)	A set of hardware and software components that provide facilities (computing, memory management, input/out, etc.) necessary to support applications.
Monotonicity	Monotonicity is the property of a variable whose value is either always increasing or always decreasing over time.

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

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

10.2 Abbreviation

Abbreviation	Meaning
AES	Advanced Encryption Standard (defined in [AES])
API	Application Programming Interface
BSP	Board support package
CA	Client Application
CC	Common Criteria
CEM	Common Evaluation Methodology
CM	Configuration Management
DES	Data Encryption Standard

Abbreviation	Meaning
DRBG	Deterministic Random Bit Generator
DRM	Digital Rights Management
EAL	Evaluation Assurance Level
ECU	Electronic control unit
EE	Execution Environment
ID	IDentifier
IVI	In-vehicle infotainment
HD	High-Definition
HDMI	High-Definition Multimedia Interface
JTAG	Joint Test Action Group
MAC	Message Authentication Code
NA	Not Applicable
NFC	Near Field Communication
NWd	Normal World (REE)
OEM	Original Equipment Manufacturer.
OMTP	Open Mobile Terminal Platform
OS	Operating System
OSP	Organisational Security Policy
OTP	One-Time Password
PCB	Printed Circuit Board
PP	Protection Profile
RAM	Random Access Memory
REE	Rich Execution Environment
RFC	Request For Comments; may denote a memorandum published by the IETF
ROM	Read Only Memory
RSA	Rivest / Shamir / Adleman asymmetric algorithm
SAR	Security Assurance Requirement
SD	Secure Driver (Privileged Trusted Application in Kinibi)
SIP	Silicon Provide. Integrates Kinibi in the SoC BSP.
SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SP	Service Provider; Develops SP TA.
SoC	System-on-Chip

Abbreviation	Meaning
SPD	Security Problem Definition
SSL	Secure Sockets Layer
ST	Security Target
SWd	Secure World (TEE)
TA	Trusted Application
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFI	TSF Interface
USB	Universal Serial Bus
VPN	Virtual Private Network

10.1 Associated references

Table 10-1: Normative References

Standard / Specification	Description	Ref
GP_PRO_023	GlobalPlatform Device Technology TEE Certification Process (last applicable version)	[TEE CP]
GPD_GUI_044	GlobalPlatform Device Technology TEE Evaluation Methodology (last applicable version)	[TEE EM]
GPD_SPE_007	GlobalPlatform Device Technology TEE Client API Specification v1.0	[TEE CLIENT 1.0]
GPD_EPR_028	GlobalPlatform Device Technology TEE Client API Specification v1.0 Errata and Precisions v2.0	[TEE CLIENT E 2.0]
GPD_SPE_010	GlobalPlatform Device Technology TEE Internal Core API Specification v1.0	[TEE CORE 1.0]
GPD_EPR_017	GlobalPlatform Device Technology TEE Internal Core API Specification v1.0 Errata and Precisions v1.0	[TEE CORE E 1.0]
	GlobalPlatform Device Technology TEE Internal Core API Specification v1.0 Errata and Precisions v3.0	[TEE CORE E 3.0]
	TEE Initial Configuration Test Suite v1.1.0.1	[TEE ICTS 1.1.0.1]

Standard / Specification	**Description**	**Ref**
GPD_SPE_021	TEE Protection Profile PP-configuration composed of the base Protection Profile only v1.2.1	[TEE PP 1.2.1]
GPD_SPE_009	GlobalPlatform Technology TEE System Architecture v1.2	[TEE SA 1.2]
IETF RFC 2119	Key words for use in RFCs to Indicate Requirement Levels	[RFC 2119]
CCMB-2017-04-001	Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1. Revision 5. April 2017.	[CC1]
CCMB-2017-04-002	Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1. Revision 5. April 2017.	[CC2]
CCMB-2017-04-003	Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements. Version 3.1. Revision 5. April 2017.	[CC3]