Public

ArcaShield Platform

Class ASE

Revision 0.2

June 2025

Security Target Lite

Revision History

Revision No.	Date	Description
0.0	26th May 2025	Creation
0.1	28th May 2025	Table 1-1, 1-2 and 1-6 are updated. The chapter 6.1.1.10 is updated.
0.2	3rd June 2025	Table 1-1 and 1-6 is updated.

Contents

FIGURES

TABLES

1 INTRODUCTION

This Security Target Lite introduces the ArcaShield Platform which satisfies the requirements of the Common Criteria EAL5 augmented with AVA_VAN.5 and ALC_DVS.2. It defines the security rules of the Target of Evaluation (TOE) and describes the environment it operates in.

The composite evaluation is based on assessment and analysis of the ArcaShield Platform on the S3SSE2A HW.

1.1 Document Overview

This document is divided into the following sections:

- Chapter 1: **Introduction** provides an introduction to this document and an overview of the TOE.
- Chapter 2: **Conformance Claim** includes the conformance claims.
- Chapter 3: **Security Problem Definition** provides information on the security problems related to the TOE. This section also defines the set of threats to be addressed either by the countermeasures implemented in the TOE hardware, the TOE software, or through environmental controls.
- Chapter 4: **Security Objectives** defines the security objectives for the TOE, operational environment, and the rationale of the security objectives. This explicitly demonstrates that the security objectives of the information technology satisfy the policies and threats. All policies and threats are mapped to the respective arguments.
- Chapter 5: **Extended Components** contains the details of the Extended Component Families introduced within this ST.
- Chapter 6: **Security Requirements** contains the security functional requirements and assurance requirements derived from the Common Criteria Part 2 [CC2], and Part 3 [CC3] to be adhered to couple with the rationale of the security functional requirements. The section then explains how the set of requirements are relative to the objectives, and one or more component requirements address the security objectives. The relevant arguments support each objective.
- Chapter 7: **Statement of Compatibility** shows that there are no contradictions or incompatibilities between this ST and the Platform ST [ICST].
- Chapter 8: **TOE Summary Specification** contains the summary of the TOE specifications.
- Chapter 9: **Acronyms** provides information on the acronyms.
- Chapters 10: **Glossary** provides information on the glossary.
- Chapters 11: **Reference Documents** provides the related references.

1.2 Security Target Lite Reference

Security Target Lite and associated evaluation are completely defined by information located in the following table.

Table 1-1 Security Target Lite Reference

ST Lite Title	Security Target Lite – ArcaShield Platform
ST Lite Author	Samsung Electronics Co., Ltd.
ST Lite Version	V0.2
Release date	3rd June 2025
Certification Body	ANSSI
Evaluation Scheme	France

The security target Lite describes the following:

- Target of Evaluation (TOE) as well as the TOE components, the components in the TOE environment, the product type and its life cycle.
- TOE security environment as well as the assets to be protected, threats to be countered by the TOE and the operational environment during the development and the active phase of the platform.
- TOE security objectives and of its operational environment in terms of integrity and confidentiality of sensitive information.
- The organizational security policies and the assumptions.
- The security requirements that includes the TOE functional requirements, the TOE assurance requirements and the security requirements for the environment.
- The TOE summary specification describes the security functions that meet the TOE security requirements.
- The composite product including an underlying IC certified according to [ICST].

1.3 TOE Reference

ArcaShield Platform is the Target of Evaluation (TOE) of this Security Target Lite. Following Table provides the TOE references.

Table 1-2 TOE Reference

TOE Name	ArcaShield Platform
TOE Version	0003040E
TOE Developer	Samsung Electronics Co., Ltd.
IC Reference	<S3SSE2A_20240430>
IC Certificate	<ANSSI-CC-2024/26>

1.4 TOE Overview

Smart cards are used as data carriers that are secure against forgery and tampering as well as personal, highly reliable, small size devices capable of replacing paper transactions by electronic data processing. Data processing is performed by a piece of software embedded in the smart card chip, called an application.

The TOE, one such smart card in the form of an embedded Secure Element (eSE), is a tamper resistant component used in a device to provide security, confidentiality and multiple application execution environments.

In an embedded environment, it is challenging to address diverse security requirements through existing security applications. One of the primary challenges in meeting requirements is validating that the security features can protect against external attacks. For instance, the encryption module should be validated for its confidentiality and the low possibility of key leakage. Or there is random number generator, the generator should be proven the randomness of the nonce from generator.

Another challenge is that there is intensive resource of security applications maintenance and management, even when the security feature is developed on existing security applications. When there are changes to the encryption module, it should be necessary to re-prove the confidentiality of module by this change.

Lastly, since sensitive data is not physically separated, additional protection against leakage by attackers is required. There is a possibility that an attacker can exfiltrate the sensitive data when they are aware of the right to access the sensitive data.

To overcome these challenges, a secure platform that meets various security requirements has been implemented on a eSE(embedded Secure Element). The secure platform provides a secure execution environment.

TOE provides security features and security execution environments to respond to security requirements similar to Java card OS. TOE also provides C language SDK to develop applications with TOE's security environment.

Applications that can run on TOE perform sensitive data storage or encryption/decryption operations. The most typical applications that can be present on the TOE are:

- ✓ Authentication application for running sensitive features

- ✓ Secure storage application for secret
- ✓ Attestation application with pre-stored certificates
- ✓ Key management application

TOE supports the security features as follows:

- ✓ Secure booting through verifying the integrity of the kernel and applications
- ✓ Memory protection by firewall to prevent the leakage of sensitive data
- ✓ Application isolation to each other for preventing conflicts

The main goal of TOE is to provide a secure and robust execution environment that meets various security requirements. TOE protects sensitive data and application execution environments from unauthorized access and prevents the leakage of sensitive data.

1.4.1 TOE Type

The TOE is a secure platform, native operating system, involved hardware crypto module, software kernel and system applications without either user applications.

1.5 TOE Description

The Target of Evaluation (TOE) is a native secure platform into a mobile handset or any other mobile device. The TOE consists of the related embedded software and the firmware combined with the underlying hardware and is a multi-application platform.

1.5.1 TOE Boundaries

1.5.1.1 TOE Physical Boundaries

The S3SSE2A IC is a tamper-resistant chip in the package.
For the present evaluation, the TOE physical boundaries encompass the S3SSE2A IC with ArcaShield Platform software. Any other item is outside the scope of the evaluation.

1.5.1.2 TOE Logical Boundaries

TOE logical boundaries are delimited (dash line) in Figure 1-1 .

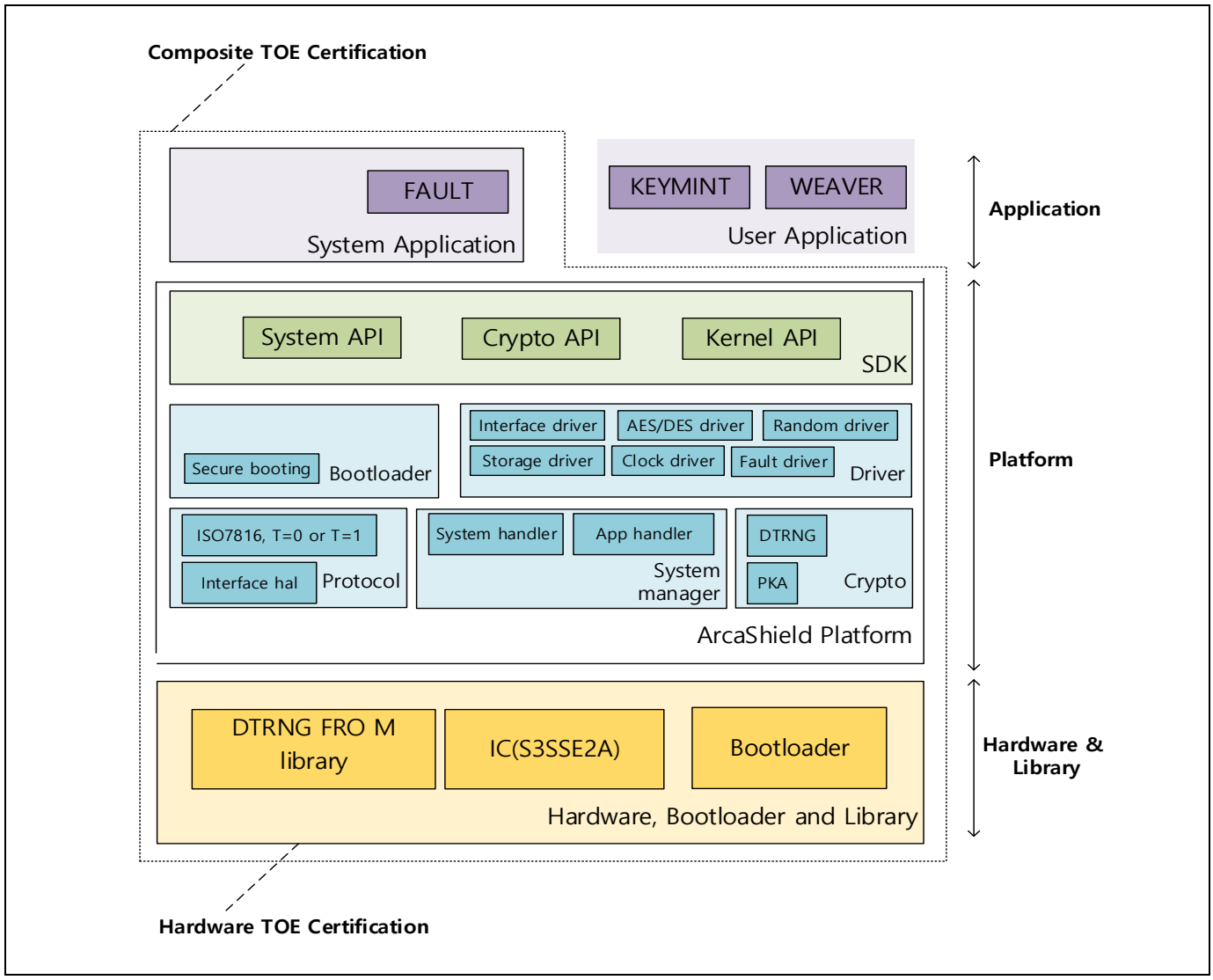

Figure 1-1 Logical Boundaries

The architecture is decomposed in three layers:

- The hardware layer composed of the S3SSE2A integrated circuit
- The ArcaShield platform layer, which is the operating system of the product
- The System Application layer.

1.5.2 ArcaShield Platform Description

From a logical point of view, the TOE comprises the following components:

- Bootloader
- Driver
- Protocol
- Crypto
- System Manager

- SDK
- System Application

Each Subsystem has some components for security service and TOE execution. The evaluation results of S3SSE2A, HW on Figure 1-1, are reused for the current evaluation.

1.5.2.1 Bootloader

Bootloader is the component to be executed at first. When the TOE operates, the bootloader is executed at first and waits for command. The bootloader supports secure booting features.

1.5.2.1.1 The secure booting

The secure booting is the security feature that the bootloader verifies the integrity of components before running components. The components consists of a header which has the information of component and a code that is executed to run features. A header has the RSASSA-2048-PKCS-V1.5., which meets the following: *[RFC2313]*, signature for header contents to authenticate itself. A code has the SHA2-256 value for code contents to authenticate itself. The bootloader verifies a header and a code using these authentication values. The bootloader separates the 'Crypto' component and other component.

The means of TOE component integrity verification for each level is as follows:

Table 1-3 Verification for TOE component

Level	Crypto component		Kernel components		Support
	Header	Code	Header	Code	
1	RSA signature	SHA256	RSA signature	SHA256	**Supported by TOE**

Note1: The Crypto component is the Subsystem "Crypto" in Figure 1-1. Other components are the Subsystem "Driver", "Protocol" and "System manager".

After jumping to the System Handler, TOE can select and execute an application by the host system's command. Before TOE selects the application, TOE checks the integrity of application. Checking the integrity is same like bootloader, the integrity of header and code shall be checked.

Table 1-4 Verification for Application

Level	Application		Support
	Header	Code	
1	RSA signature	crc32	**Supported by TOE**

When the application verification is success, TOE can execute the application.

There is the difference of the integrity method between CRC32 of "Code of Application in Table 1-4 Verification for Application" and SHA256 of "Code of Kernel components in Table 1-3 Verification for TOE component".

1.5.2.2 Driver

The Driver component consists of driver modules related hardware interface, storage, random and any other hardware components.

1.5.2.2.1 Interface Driver

Interface Driver provides the features for communicating with host system. Interface Driver supports the send/receive/synchronize features for communication to each supported protocols. Interface Driver supports following protocols.

- ✓ SPI Protocol

Interface Driver works with 'protocol' component for communication with host system. 'Protocol' component parses the data according to protocol manner. Interface Driver sets the data and configuration to the hardware register for communication.

1.5.2.2.2 Storage Driver

Storage Driver supports the feature to read/write/delete data on flash storage area securely. Storage Driver maps the physical address of flash storage area to the RAM variable.

1.5.2.2.3 Clock Driver

Clock Driver supports the variable clock. Clock Driver supports the feature for managing clock of hardware components. Clock Driver enables and disables the hardware component's clock when the hardware component is run. The Driver can also control the frequency of clock for core and Crypto LIB hardware. Clock Driver offers the variable clock on/off feature for security.

1.5.2.2.4 AES/DES Driver

AES Driver provides the security feature which is the AES encryption/decryption calculation for other component and application layer. The application layer does not use the AES driver directly, it must go through the applicative SDK in order to call a kernel service. AES Driver provides following cryptography mechanisms:

- ✓ AES with 128, 192 and 256 bits key sizes
- ✓ Supported cipher mode : ECB, CBC, CTR
- ✓ Supported padding scheme : PKCS7

AES Driver protects the key value and the intermediate encryption/decryption result value against side channel attack. AES Driver also prevents the malfunction by fault injection attack.

DES Driver provides the security feature which is the Triple DES encryption/decryption calculation for other component and application layer. The application layer does not use the DES driver directly, it must go through the applicative SDK in order to call a kernel service. DES Driver provides following cryptography mechanisms:

- ✓ Triple DES with 192 bit key sizes
- ✓ Supported cipher mode : ECB, CBC, CTR
- ✓ Supported padding scheme : PKCS7

DES Driver protects the key value and the intermediate encryption/decryption result value against side channel attack. DES Driver also prevents the malfunction by fault injection attack.

1.5.2.2.5 Random Driver

Random Driver supports the BPRNG (Binary Pseudo Random Number Generator) feature which is used for only security countermeasures such as random masking. This shall not be used for crypto purpose such as key generation. This feature is faster than DTRNG (Digital True Random Number Generator) on 'Crypto' component.

1.5.2.2.6 Fault Driver

Fault Driver is the security feature to progress when the fault occurs. Fault is occurred by hardware malfunction or fatal attack. TOE is able to configure 2 handler functions for the fault. One handler function is configured to respond to fault. Another handler function transits to a recovery state when a fault is occurred.

1.5.2.3 Protocol

Protocol is the component for communication with host system. Protocol component works with 'Interface driver' in 'driver' component. Protocol component is initialized with the bootloader.

1.5.2.3.1 ISO7816, T=0 or T=1

'ISO7816, T=0 or T=1' module offers the feature for ISO7816 communication method. The method which is used by TOE is decided when TOE is built. TOE uses the ISO7816 T=1 communication method by default. The components of TOE communicate with host system through this module functions.

'ISO7816, T=0 or T=1' module check whether the received data from host system complies with the communication method used in TOE and constructs the data which is sent to host system in accordance with the communication method.

1.5.2.3.2 Interface hal

'Interface hal' module offers the functions to handle the physical communication component through 'Interface driver'. The functions of 'Interface hal' is provided to 'ISO7816, T=0 or T=1' module and used to send/receive the data through the method chosen by TOE. The method is SPI protocol. TOE decides the method when TOE is built.

1.5.2.4 Crypto

Crypto component provides the cryptography mechanisms to other component and application layer through SDK.

1.5.2.4.1 DTRNG

DTRNG (Digital True Random Number Generator) offers the security feature which provides the random nonce. The nonce is securely generated against external attacks and ensures randomness.

1.5.2.4.2 PKA

PKA module provides security feature which is the cryptographic services to the other components and the application layer through SDK. The offered features by PKA module are prevented from the external attack like the fault injection attack or the side channel attack.

The cryptographic mechanisms are provided as follows:

1. Hash Generation
   - ✓ SHA variants –SHA256, SHA384 and SHA512
   - ✓ Support Virtual Hash feature : SHA256, SHA384 and SHA512
2. Asymmetric cryptography
   - ✓ RSA2048
3. Key Exchange
   - ✓ Elliptic Curve Diffie-Hellman (DH) Key Exchange
4. Padding Scheme
   - ✓ PKCS1 Version 1.5
   - ✓ PKCS1 PSS
   - ✓ OAEP

Note 1: The ECC functions are supported for any valid elliptic curves over prime fields of sizes from 256 to 512 bits. However, only the proven standard curves listed below are in the scope of this evaluation:

1. [NIST curves]: Curve P-P-256, P-384 and P-521
2. [SEC-recommended curves]: secp256k1, secp256r1, secp384r1, secp521r1

Note2: The SHA variants support the feature through SHA*_init function, SHA*_update function and SHA*_final function. SHA256, SHA384 and SHA512 offer the two kinds of _update function and _final function. One SHA*_update and SHA*_final functions prevent only fault injection attack. Another SHA*_update and SHA*_final functions prevent the fault injection attack and side channel attack. Side channel attack is prevented by the virtual hash feature.

1.5.2.5 System Manager

System Manager is the component for managing the other components to progress the task. System Manager gets the APDU command first. It checks the command to identify the subsystem which processes the command. System Manager has three modules for TOE components, application layer and particular storage area.

1.5.2.5.1 System handler

System handler is started by the bootloader subsystem. Host system sends the APDU command to System handler manages whether the APDU command is executed. System handler has the context value to manage the APDU command execution.

1.5.2.5.2 App handler

App handler module manages the application execution. When the host system tends to run applications, TOE needs to receive the application initialize command. App handler receives the application initialize command and checks the application id in the command. When the application id is correct, App handler switches the privilege to execute applications and executes application initialize function defined on application. App handler also activates MPU to sandboxing application

environment (Refer to the 8.2.1). After then App handler configures the logical channel with the application which is initialized.

1.5.2.6 SDK

The SDK is the component to provide the several features to applications through APIs. This subsystem consists of three modules 'Kernel API', 'System API' and 'Crypto API'.

The Kernel API supports the non-crypto functions like clock control or storage management. It is provided to user application developers. The System API provides the additional APIs more than Kernel API module to run system application which is generated by TOE developers. The Crypto API is used for both user application and system application.

The SDK is mandatory for user applications to access the kernel and crypto features.

1.5.2.6.1 Kernel API

Kernel API provides the feature through API to application layer. This APIs are executed with application and progress on tasks according to application developer's requirement. Kernel API provides the following features in form of API:

1. Application Configuration
   - ✓ Application configuration based on iso7816_3
2. Storage management only for application data area
   - ✓ Read/Write/Delete the data on the allowed flash storage area
   - ✓ Acquire the address of the allowed flash storage area and the size of the storage area
3. Provide TOE system information
   - ✓ Acquire TOE system version /system build date
4. Fault feature by the application
   - ✓ When the application is working, the user can generate the fault through this API.
5. Provide the TOE system time
   - ✓ Acquire TOE system tick or system time
6. Provide the TOE hardware serial number to application layer
7. Calling the protocol layer to send/receive the data
   - ✓ SPI Protocol
8. Timer feature
   - ✓ Provide the timer APIs to configure manual timer
   - ✓ Provide the basic timer (millisecond, microsecond)
9. Secure memory control
   - ✓ Provide secure memory compare/copy/XOR operation/set / move API
   - ✓ Security of these function is based on memory shuffling
10. Random number generator
    - ✓ The nonce from this API can only be used on security countermeasure. It shall not be used to security sensitive value such as key generation.
11. Variable clock control
    - ✓ Enable /Disable Variable clock

1.5.2.6.2 System API

System API is used to manage the TOE through system application which is provided by TOE developers. This feature shall not be provided to user application.

1.5.2.6.3 Crypto API

Crypto API provides the security features which is cryptographic mechanisms to applications. Crypto API is provided to user applications and system applications. User applications uses these APIs to meet their security requirements. The cryptographic mechanisms are provided through Crypto API as follows:

1. Hash Generation
   - ✓ Referred to 1.5.2.4.2
2. Symmetric cryptography
   - ✓ AES / DES on ECB, CBC, CTR mode referred to 1.5.2.2.4
   - ✓ AES on GCM mode with 128, 192 and 256 bits key sizes
3. Asymmetric cryptography
   - ✓ Referred to 1.5.2.4.2
   - ✓ When getting RSA signature, user shall input the digested message.
4. Key Exchange
   - ✓ Referred to 1.5.2.4.2
5. Key Derivation
   - ✓ KBKDF (Key Based Key Derivation Function)
   - ✓ PBKDF2 (Password Based Key Derivation Function)
   - ✓ HKDF (HMAC based Key Derivation Function)
6. Authentication Code
   - ✓ HMAC with SHA256, SHA384 and SHA512
   - ✓ CMAC with AES-128, 192, and 256 bits key sizes
7. Secure Random number generator
   - ✓ Referred to 1.5.2.4.1

AES on GCM mode, Key Derivation and Authentication Code are running on SDK component. The features are applied with countermeasures to prevent the external attack, such as the side channel attack and the fault injection attack.

1.5.2.7 System Application

TOE developers offers the System Application to user. System Application is able to use the System API and Crypto API in SDK component. System Application also configures the process when the fault occurs. Fault application, another part of System Application component, offers the process of fault detecting.

1.5.2.7.1 Fault application

Fault application provides the fault handling feature. It handles situation after fault occurs.

1.5.3 TOE Identification

The TOE configuration is summarized in Table 1-6 below:

Table 1-5 TOE Configuration

Distributed Name	Version	Description	Distribution method	Type
ArcaShield Platform	0003040E	Executable Code	Enciphered hex file[1]	Software
S3SSE2A IC	0	See IC certificate	See [ICST]	Hardware
ArcaShield Platform on S3SSE2A IC	Refer to above.	The whole composite TOE	See [ICST][2]	Hardware
ArcaShield_UM_OPE	1.5	Operation Guidance documents	Pdf file[3]	Document
ArcaShield_UM_PRE	0.6	Guidance documents	Pdf file[4]	Document
SDK for Kernel API (star_kernel.lib)	0003040C	SDK library (sha256sum: 8bcd85d5b083560d91c607be8aaf9ddd6060f741d4c868f462209e7142396ca4)	.lib file	Software
SDK for Kernel API (star_secure.lib)	0003040C	SDK library (sha256sum: 70607328f423ced7e069e2fc1c21fb2e16327c8c5acfbb11abacd13110b6947e)	.lib file	Software
SDK for Crypto API (star_crypto.lib)	0003040C	SDK library (sha256sum: 82ebfc6b55ba82ef66dfc6ed5721539427374afb48a66c7125cc33cc5d54419f)	.lib file	Software

User can identify the version of ArcaShield platform and the version of S3SSE2A IC by GET INFO command. Refer to the Appendix A of [ArcaShield_UM_OPE] and section 3 of [ArcaShield_UM_PRE].

1.6 TOE Life Cycle

1.6.1 TOE Composite Life Cycle

Figure 1-2 illustrates the product life cycle phases with respect to [ICPP].

The product lifecycle is divided into the following phases

[1] Enciphered hex file is delivered by email after PGP encryption.
[2] Composite product Integration and Pre-personalization are done during wafer testing step in Phase3. The wafer injected Platfrom is delivered from Phase3 to Phase4.
[3] The pdf file is delivered by email after PGP encryption.
[4] The pdf file is delivered by email after PGP encryption.

- Phase 1 comprise the IC embedded software(Platform and Application) development.
- Phase 2 comprise IC development(IC design, IC Dedicated Software development).
- Phase 3 correspond to IC manufacturing, wafer testing and the loading of software Platform and Application components within the IC.
- Phase 4 correspond to IC packaging.
- Phase 6 is dedicated to the product personalization prior final use.
- Phase 7 is the product operational phase.

The ArcaShield Platform life cycle comprises of the following stages:

- Development

ArcaShield Platform Development is performed during Phase 1.

The deliveries of the “ArcaShield Platform” and “User Application” occurs from "Phase 1(Embedded Software Development) site" to "Phase 2(Security IC Development) site".
Regarding the Phases 1, 2, 3 and 4, the Security IC has already been certified (e.g. against [ICPP]) and hence re-evaluation is not required.

In Phase 1 case, it is the IC embedded software development site.

In Phase 2 case, it is the IC development site and is also the merge site of ArcaShield Platform and user Application.

In Phase 3(Wafer testing step) instead of Phase 5, the Composite Product Integrator, the person in charge of wafer testing. stores and pre-personalizes the ArcaShield Platform and user Application, and potentially conducts tests on behalf of the ArcaShield Platform developer.

In Phase 4 case, it is the IC packaging site. The delivery of the composite TOE (Arcashield Platform, user Application and security IC) is done from Phase4 to Phase 6. Refer to the Figure 1-2.

In Phase 5(Composite product Integration and Pre-Personalization) case, it is covered in Phase3.

In Phase 6(Personalization) case, it is not a scope of ArcaShield Platform but it is the scope of user Application.

In the ArcaShield Platform final usage phase (Phase 7), the ArcaShield Platform is embedded within the IC. The ArcaShield Platform and the product provide a full set of security functionalities that prevents product abuse by untrusted entities.

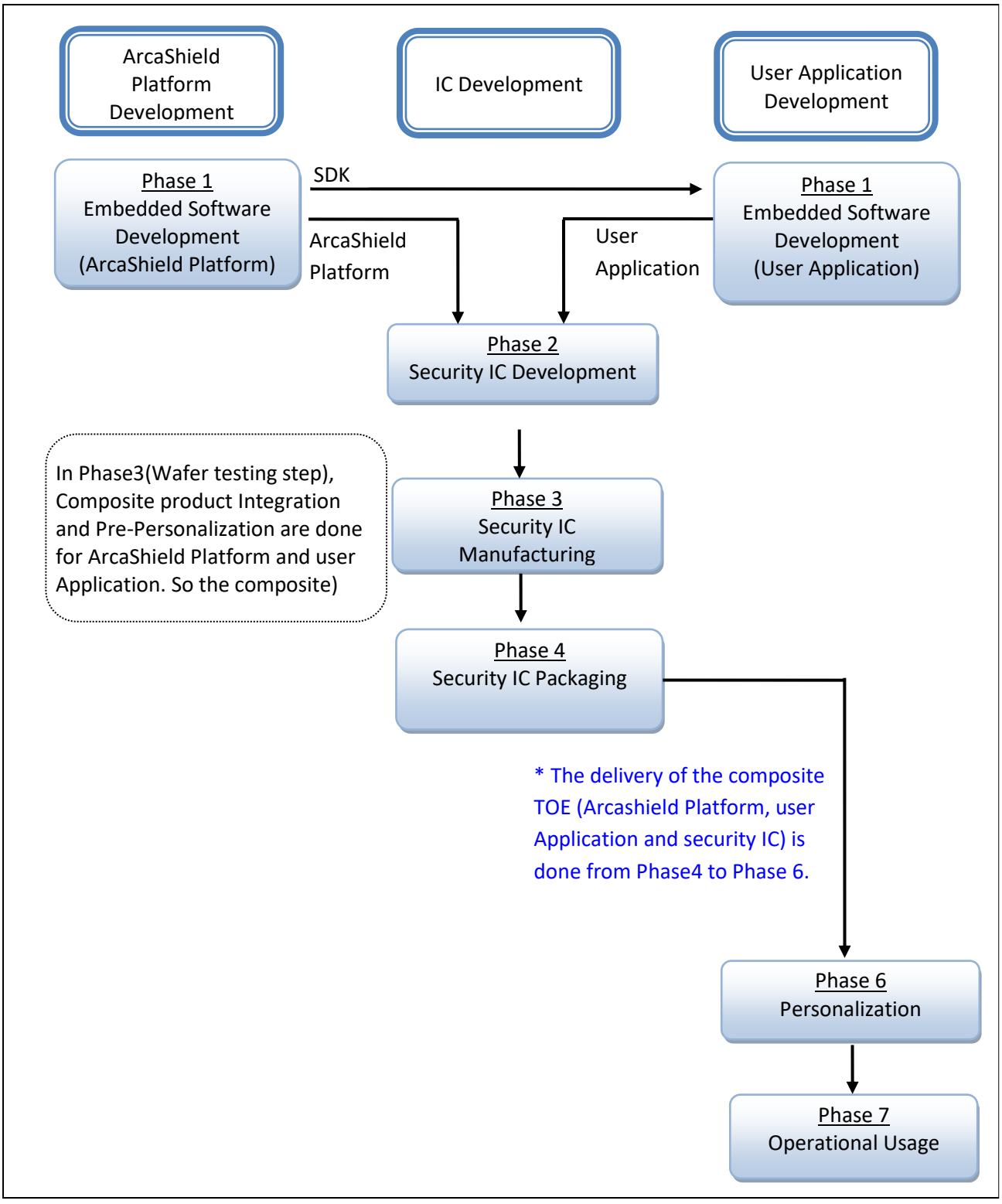

Figure 1-2 TOE Life Cycle

Table 1-6 TOE life-cycle phases

Phase	Function (phase)	Description	Actors	Location
1	ArcaShield Platform, SDK software development	Platform development & tests	**Samsung ArcaShield Platform developer** (Secure environment)	IC Development site(Phase 2) stated in the S3SSE2A certificate
2	IC development	the IC development site, merge site of ArcaShield Platform and user Application.	**Samsung IC developer** (Secure environment)	Development site(s) stated in the S3SSE2A certificate
3	IC manufacturing	Manufacturing of virgin S3SSE2A integrated circuits. Wafer testing, Composite product Integration and Pre-Personalization	**Samsung IC developer** (Secure environment)	Development site(s) stated in the S3SSE2A certificate
4	IC packaging	Module creation: IC packaging & testing	**Samsung IC developer** (Secure environment)	Development site(s) stated in the S3SSE2A certificate
6	Personalization	Personalization and final tests: personalization of the TOE and end-user applicative data	**User Application developer** (Secure environment)	User Application site
7	End-usage	End-usage for mobile phone holder The end-user accesses the OEM related services and performs secure transactions with his mobile phone.	Mobile phone	Field

1.6.2 TOE Software Components Life Cycle

1.6.2.1 ArcaSshield Life Cycle States

1.6.2.1.1 Manufacturing Phase

The following additional operations occur during the manufacturing phase:

- Initializing ArcaShield Platform.

The ArcaShield Life Cycle switches automatically to OP_READY state when the above steps are completed. Do note that the transition is irreversible.

1.6.2.1.2 OP_READY

The following operations can be performed when the ArcaShield is in the OP_READY state:

- Boot jump for executing an application
- Select an application
- Execute an application
- Manage an application or component

1.6.2.1.3 RESTRICTED

When the fault occurs defined times, the state transits from OP_READY to RESTRCTED. TOE in RESTRICTED state can be recovered to OP_READY state by authorized user. The following operations can be performed when it is in a RESTRICTED state:

- Disable the selection of applications except the System Application

- Acquire the stored log when the failure is occurred.
- Recover the state with authentication
- Check whether the state is RESTRICTED.

1.6.2.2 Application Life Cycle States

The application life cycle starts with the INSTALLED state. The application has the following states – INSTALLED.

1.6.2.2.1 INSTALLED

In the state:

- ✓ Execution of the application by the App Handler allowed.
- ✓ When the state is INSTALLED, System Handler checks the information of application and allows the update the application. This time, the state maintains INSTALLED.

1.6.2.3 System Application Life Cycle States

System Application is provided with TOE platform. The system application has the following states – INSTALLED.

1.6.2.3.1 INSTALLED

In the state:

- ✓ Execution of the application by the App Handler allowed.
- ✓ When the state is INSTALLED, System Handler checks the information of application and allows the update the application. This time, the state maintains INSTALLED.

2 CONFORMANCE CLAIM

This Security Target and the TOE it describes are fully compliant with Common Criteria 3.1R5.

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

- CC Part 2 [CC2] extended.
- CC Part 3 [CC3] conformant.

The methodology to be used for the evaluation is described in the "Evaluation methodology" of the Common Criteria Standard, April 2017, Version 3.1 Revision 5 [CEM] with an Evaluation Assurance Level of EAL5 augmented with the assurance components ALC_DVS.2 and AVA_VAN.5.

The Composite evaluation methodology [JIL_COMP] is used to perform the associated evaluation and hence IC relevant information is not repeated in the current ST.

2.1 Rationale of the Conformance Claim

No Protection Profile is claimed.

Note however that this ST is inspired by Closed Java Card System Protection Profile [JCSPP]. The Security Problem Definition, Security Objectives and Security Requirements are taken from this PP and modified or added/removed in order to adapt these elements to the current TOE.

3 SECURITY PROBLEM DEFINITION

This chapter defines the security problems to be addressed by the TOE and the operational environment of the TOE.

One of the characteristics of the eSE is that several entities are represented inside the platform as follows:

- **Platform Provider –** The owner of ArcaShield platform. Platform Provider develops the platform. Platform Provider provides the platform and SDK (Kernel API and Crypto API) to Application Provider.
- **Application Provider –** The entity or institution responsible for the applications and their associated services. An Application Provider can be a customer company or a third party service provider. Application Provider provides the normal applications excluded the System Application.

3.1 Assets

Assets are the security-relevant elements to be directly protected by the TOE. Since various parties are involved during the initial stages of the smart card product lifecycle, it is imperative to maintain the confidentiality of assets in terms of software as well as the human resources. For more details about the threats, refer section Threats.

The TOE protects the following assets:

- System related assets
- User application related assets.

3.1.1 Runtime Environment Assets

The following table shows the assets.

Table 3-1 Assets

System related Assets	**D.SYS_CODE**	Platform code
	D.SYS_DATA	Platform level cryptographic keys (for Platform Provider authentication, secure boot) Internal platform data – platform life-cycle (INSTALLED), fault detection related data (fault counter / dumped call stack value when fault is occurred), execution mode related data (non-RESTRICTED mode / RESTRICTED mode) and root of trust data (public key hash value stored in OTP)

	D.SYS_CONTENT	Content related context (for all user applications), registry of loaded application (application status, application life-cycle), Code and Data memory areas configuration
	D.SYS_CURRENT_CONTEXT	Execution context of current application (the flags and attributes related to current application, current application sandbox configuration)
User Application related Assets	**D.APP_CODE**	User application code
	D.APP_C_DATA	User application permanent and volatile data to be protected in confidentiality
	D.APP_I_DATA	User application permanent and volatile data to be protected in integrity
	D.APP_STATE	State of the application

3.2 Threats

This section describes the threats to the assets against the specific protection within the TOE or its required environment. Several groups of threats are distinguished according to the mode of attack. The components of the TOE inspire the classification to counter each threat.

The following threat agents are considered:

Attacker: A human or a process acting on his behalf located outside the TOE. The main goal of the attacker is to modify or corrupt sensitive information.

3.2.1 Threats against Confidentiality an Integrity of data

- **T.APP_C_DATA –** The attacker forces an application to disclose data belonging to another application.
  - Attacker capabilities: attacker that can load a malicious application
  - Directly threatened assets: D.APP_C_DATA, D.SYS_CODE, D.APP_CODE, D.SYS_DATA

- **T.APP_I_DATA –** The attacker forces an application to modify data belonging to another application.
  - Attacker capabilities: attacker that can load a malicious application
  - Directly threatened assets: D.APP_I_DATA, D.SYS_CODE, D.APP_CODE, D.APP_STATE.

- **T.APP_CODE –** The attacker forces an application to modify code belonging to another application.
  - Attacker capabilities: attacker that can load a malicious application
  - Directly threatened assets: D.APP_CODE.

- **T.SYS_DATA –** The attacker forces an application to modify data belonging to the platform.
  - Attacker capabilities: attacker that can load a malicious application
  - Directly threatened assets: D.SYS_DATA

- **T.SYS_CODE –** The attacker forces an application to modify code belonging to the platform.
  - Attacker capabilities: attacker that can load a malicious application
  - Directly threatened assets: D.SYS_CODE

3.2.2 Attack means related threats

- **T.PHYSICAL –** An attacker may perform physical probing, physical modification or apply environmental stress in order to (i) deactivate or modify security features or functions of the TOE, (ii) circumvent, deactivate or modify security function, (iii) to disclose confidential TSF data, (iv) modify the TSF data
  - Attacker capabilities: attacker that can load an application and that can apply the pysical manipulation and /or environmental stress
  - Directly threatened asset(s): all assets

- **T.INFORMATION_LEAKAGE –** An attacker may exploit information which is leaked from the TOE during its usage in order to disclose confidential TSF data. The information leakage may be inherent in the normal operation or caused by the attacker.
  - Attacker capabilities: attacker that can load an application and that can listen or observe various hidden communication channels
  - Directly threatened assets: confidentiality related assets

3.3 Organizational Security Policies (OSPs)

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

3.3.1 OSP.INTEGRATION_CONFIGURATION

Integration and configuration of the TOE by the device manufacturer shall rely on guidelines defined by the TOE provider, and state all the security requirements for the device manufacturer issued from the TOE evaluation.

3.3.2 OSP.SECRETS

Generation, storage, distribution, destruction, injection of secret data in the TOE or any other operation performed outside the TOE shall enforce integrity and confidentiality of this data. This applies to secret data injected before end-usage phase (such as the root of trust which is used only for verifying the integrity of the kernel ("secure booting") and the integrity of applications) or during the end-usage phase (such as cryptographic private or symmetric keys, confidential data).

3.3.3 OSP.CRYPTO

The TOE provides secure hardware based cryptographic services for the Applications.

3.4 Assumptions

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

3.4.1 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 guidelines (e.g. user and administrator guidance, installation documentation, personalization guide). It is also assumed that the persons responsible for the application of the procedures contained in the guides, and the persons involved in delivery and protection of the product have the required skills and are aware of the security issues.

3.4.2 A.APP_DEVELOPMENT

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

- Applications must not disclose any sensitive data to the Host System.

4 SECURITY OBJECTIVES

4.1 Security Objectives for the TOE

Table 4-1 Objectives for the TOE

Execution	**O.ID**	The TOE shall uniquely identify every subject (application) before granting it access to any service or resource.
	O.FIREWALL	The TOE shall ensure controlled sharing of data owned by different applications
	O.OPERATE	The TOE must ensure continued correct operation of its functionalities (code execution, API, services)
	O.RESOURCES	The TOE shall control the availability of resources for the applications
API and services	**O.ALARM**	The TOE shall provide appropriate feedback information upon detection of a potential security violation.
	O.KEY_MNGT	The TOE shall provide a means to securely manage cryptographic keys. This concerns the correct distribution, access and destruction of cryptographic keys.
	O.CRYPTO	The TOE shall provide a means to cipher sensitive data and generate/verify signature for applications in a secure way. In particular, the TOE must support cryptographic algorithms consistent with cryptographic usage policies and standards.
	O.RNG	The TOE shall ensure the cryptographic quality of random number generation. For instance, random numbers shall not be predictable and shall have sufficient entropy. The TOE shall ensure that no information about the produced random numbers is available to an attacker since they might be used for instance to generate cryptographic keys.

4.2 Security Objectives for the Operational Environment

4.2.1 OE.INTEGRATION_CONFIGURATON

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

4.2.2 OE.PROTECTION_AFTER_DELIVERY

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

4.2.3 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 TOE shall enforce integrity and confidentiality of these data.

4.2.4 OE.APP_DEVELOPMENT

Application developers shall comply with the Application development guidelines set by the TOE provider. In particular, Application developers shall apply the following security recommendations during the development of the Secure Applications:

- Application does not assume that it always receives normal commands.
- Application does not assume that it receives commands only from the authorized Host System.
- Application does not disclose any sensitive data to the Host System.

4.3 Security Objectives Rationale

4.3.1 Threats

4.3.1.1 T.APP_C_DATA

T.APP_C_DATA: The combination of the following objectives ensures protection against disclosing data belonging to another application:

- O.FIREWALL: The TOE shall ensure controlled sharing of data owned by different applications
- O.OPERATE: The TOE must ensure continued correct operation of its functionalities (code execution, API, services)
- O.KEY_MNGT: The TOE shall provide a means to securely manage cryptographic keys. This concerns the correct generation, distribution, access and destruction of cryptographic keys.
- O.CRYPTO: The TOE shall provide a means to cipher sensitive data and generate/verify signature for applications in a secure way. In particular, the TOE must support cryptographic algorithms consistent with cryptographic usage policies and standards.
- O.RNG: The TOE shall ensure the cryptographic quality of random number generation. For instance, random numbers shall not be predictable and shall have sufficient entropy. The TOE shall ensure that no information about the produced random numbers is available to an attacker since they might be used for instance to generate cryptographic keys.

4.3.1.2 T.APP_I_DATA

T.APP_I_DATA: The combination of the following objectives ensures protection against modifying data belonging to another application:

- O.FIREWALL: The TOE shall ensure controlled sharing of data owned by different applications
- O.OPERATE: The TOE must ensure continued correct operation of its functionalities (code execution, API, services)
- O.CRYPTO: The TOE shall provide a means to cipher sensitive data and generate/verify signature for applications in a secure way. In particular, the TOE must support cryptographic algorithms consistent with cryptographic usage policies and standards.

4.3.1.3 T.APP_CODE

T.APP_CODE: The combination of the following objectives ensures protection against modifying code belonging to another application.

- O.ID: The TOE shall uniquely identify every subject (application) before granting it access to any service or resource.
- O.FIREWALL: The TOE shall ensure controlled sharing of data owned by different applications
- O.OPERATE: The TOE must ensure continued correct operation of its functionalities (code execution, API, services)

4.3.1.4 T.SYS_DATA

T.SYS_DATA: The combination of the following objectives ensures protection against modifying data belonging to platform.

- O.FIREWALL: The TOE shall ensure controlled sharing of data owned by different applications
- O.OPERATE: The TOE must ensure continued correct operation of its functionalities (code execution, API, services)
- O.KEY_MNGT: he TOE shall provide a means to securely manage cryptographic keys. This concerns the correct generation, distribution, access and destruction of cryptographic keys.
- O.CRYPTO: The TOE shall provide a means to cipher sensitive data and generate/verify signature for applications in a secure way. In particular, the TOE must support cryptographic algorithms consistent with cryptographic usage policies and standards.
- O.RNG: The TOE shall ensure the cryptographic quality of random number generation. For instance, random numbers shall not be predictable and shall have sufficient entropy. The TOE shall ensure that no information about the produced random numbers is available to an attacker since they might be used for instance to generate cryptographic keys.

4.3.1.5 T.SYS_CODE

T.SYS_CODE: The combination of the following objectives ensures protection against modifying code belonging to platform.

- O.FIREWALL: The TOE shall ensure controlled sharing of data owned by different applications
- O.OPERATE: The TOE must ensure continued correct operation of its functionalities (code execution, API, services)

4.3.1.6 T.PHYSICAL

T.PHYSICAL: The combination of the following objectives ensures protection against physical probing, physical modification or apply environmental stress in order to (i) deactivate or modify security features or functions of the TOE, (ii) circumvent, deactivate or modify security function, (iii) to disclose confidential TSF data, (iv) modify the TSF data

- O.ALARM: The TOE shall provide appropriate feedback information upon detection of a potential security violation.
- O.RESOURCES: The TOE shall control the availability of resources for the applications

4.3.1.7 T.INFORMATION_LEAKAGE

T.INFORMATION_LEAKAGE: The combination of the following objectives ensures protection against exploiting information which is leaked from the TOE during its usage in order to disclose confidential TSF data.

- O.ALARM: The TOE shall provide appropriate feedback information upon detection of a potential security violation.
- O.RESOURCES: The TOE shall control the availability of resources for the applications

4.3.2 Organizational Security Policies (OSPs)

4.3.2.1 OSP.INTEGRATION_CONFIGURATION

The objective OE.INTEGRATION_CONFIGURATION directly covers this OSP.

4.3.2.2 OSP.SECRETS

The objective OE.SECRETS directly covers this OSP.

4.3.2.3 OSP.CRYPTO

The objective O.CRYPTO directly covers this OSP.

4.3.3 Assumptions

4.3.3.1 A.PROTECTION_AFTER_DELIVERY

The objective OE.PROTECTION_AFTER_DELIVERY directly covers this assumption.

4.3.3.2 A.APP_DEVELOPMENT

The Objective OE.APP_DEVELOPMENT directly covers this assumption.

4.4 Security Objectives Coverage

The following tables provide a mapping of security objectives to the environment defined by the threats, policies and assumptions, illustrating that each security objective covers at least one threat and that each threat is countered by at least one objective, assumption or policy.

4.4.1 Mapping of Security Objectives to Threats

Table 4-2 Mapping Security Objectives to Threats

	T.APP_C_DATA	T.APP_I_DATA	T.APP_CODE	T.SYS_DATA	T.SYS_CODE	T.PHYSICAL	T.INFORMATION_LEAKAGE
O.ID			X				
O.FIREWALL	X	X	X	X	X		
O.OPERATE	X	X	X	X	X		
O.RESOURCES						X	X
O.ALARM						X	X
O.KEY_MNGT	X			X			
O.CRYPTO	X	X		X			
O.RNG	X			X			

4.4.2 Mapping of Security Objectives to Policies and Assumptions

Table 4-3 Mapping of Security Objectives to OSPs and Assumptions

	OSP.INTEGRATION_CONFIGURATION	OSP.SECRETS	OSP.CRYPTO	A.PROTECTION_AFTER_DELIVERY	A.APP_DEVELOPMENT
OE.INTEGRATION_CONFIGURATION	X				
OE.SECRETS		X			
O.CRYPTO			X		
OE.PROTECTION_AFTER_DELIVERY				X	
OE.APP_DEVELOPMENT					X

5 EXTENDED COMPONENTS DEFINITION

The TOE is part 2 extended. Extended requirements are identified as "Common Criteria Part 2 extended".

5.1 Definition of the Family FCS_RNG

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.

5.2 FCS_RNG Generation of random numbers

Family behavior

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

Component leveling:

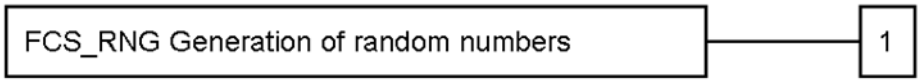

Figure 5-1 FCS_RNG Generation of Random Numbers

FCS_RNG.1 Generation of random numbers requires that random numbers meet a defined quality metric.

Management: FCS_RNG.1

There are no management activities foreseen.

Audit: FCS_RNG.1

There are no auditable events foreseen.

FCS_RNG.1 Random number generation

Hierarchical to: No other components.

Dependencies: No dependencies.

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

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

Application Note:

A physical random number generator (RNG) produces the random number by a noise source based on physical random processes. A non-physical true RNG uses a noise source based on non-physical random processes like human interaction (key strokes, mouse movement). A deterministic RNG uses a random seed to produce a pseudorandom output. A hybrid RNG combines the principles of physical and deterministic RNGs.

6 SECURITY REQUIREMENTS

The followings explain about subject, object or operation.

- Platform: Platform is the ArcaShield Platform.
- User Application: The application developed by Application Provider
- Physical Resources: Stored data or code on RAM or FLASH memory
- Code Execution: Execute the code stored on FLASH memory by Platform or Applications
- Access to Data: Read or write the data on RAM or FLASH by Platform or Applications
- Permissions: Platform allows Application to do Code Execution or Access to Data which is allowed for Application.

6.1 Security Functional Requirements

This section states the security functional requirements for the TOE.

For requirements are arranged into several groups corresponding to several usage categories.

The operations on SFRs are accented with the following typographic conventions:

- Selection and assignment: italic font
- Refinement: Application note below the SFR definition
- Iteration: **/name** notation after the standard SFR name

6.1.1 Related to platform API and services

6.1.1.1 FCS_RNG.1/API Generation of random numbers

Hierarchical to: No other components.

Dependencies: No dependencies.

FCS_RNG.1.1/API The TSF shall provide a *physical*[5] random number generator that implements: *Random padding, Zero-knowledge proof, Password generation*[6].

FCS_RNG.1.2/API The TSF shall provide *bits, 16-bit per numbers*[7] that meet *AIS31 PTG.2*[8].

Application Note: This SFR concerns the RNG proposed to user applications as a service/API. The TSF fulfils some but not all the necessary rules to comply with the guide, "Régles et recom-

[5] [selection: *physical, non-physical true, deterministic, hybrid*]
[6] [assignment: *list of security capabilities*]
[7] [selection: *bits, octets of bits, numbers [assignment: format of the numbers]*]
[8] [assignment: *a defined quality metric*]

mandations concernant le choix et le dimensionnement des mécanismes cryptographiques Version 2.04", regarding random numbers generators (RNG). The composite product's RNG will comply with the guide only when all the rules of §2.4 "Génération d'aléa cryptographique" of the guide are addressed. In particular, a cryptographic post-processing must be implemented by the Application Developer.

6.1.1.2 FCS_CKM.4/API/CMAC Cryptographic key destruction

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4.1/API/CMAC The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method *zeroing*[9] that meets the following: *No Standard*[10].

Application Note: This SFR concerns the destruction of temporal key storage cryptographic operation.

6.1.1.3 FCS_CKM.4/API/HMAC Cryptographic key destruction

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4.1/API/HMAC The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method *zeroing*[11] that meets the following: *No Standard*[12].

Application Note: This SFR concerns the destruction of temporal key storage cryptographic operation.

6.1.1.4 FCS_CKM.4/API/KDF Cryptographic key destruction

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4.1/API/KDF The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method *zeroing*[13] that meets the following: *No Standard*[14].

Application Note: This SFR concerns the destruction of temporal key storage cryptographic operation.

[9] [assignment: *cryptographic key destruction method*]
[10] [assignment: *list of standards*]
[11] [assignment: *cryptographic key destruction method*]
[12] [assignment: *list of standards*]
[13] [assignment: *cryptographic key destruction method*]
[14] [assignment: *list of standards*]

6.1.1.5 FCS_CKM.4/API/RSA Cryptographic key destruction

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4.1/API/RSA The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method *zeroing*[15] that meets the following: *No Standard*[16].

Application Note: This SFR concerns the destruction of temporal key storage cryptographic operation.

6.1.1.6 FCS_CKM.4/API/ECDH Cryptographic key destruction

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4.1/API/ECDH The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method *zeroing*[17] that meets the following: *No Standard*[18].

Application Note: This SFR concerns the destruction of temporal key storage cryptographic operation.

6.1.1.7 FCS_CKM.4/API/AES Cryptographic key destruction

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4.1/API/AES The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method *zeroing*[19] that meets the following: *No Standard*[20].

Application Note: This SFR concerns the destruction of temporal key storage cryptographic operation.

[15] [assignment: *cryptographic key destruction method*]
[16] [assignment: *list of standards*]
[17] [assignment: *cryptographic key destruction method*]
[18] [assignment: *list of standards*]
[19] [assignment: *cryptographic key destruction method*]
[20] [assignment: *list of standards*]

6.1.1.8 FCS_CKM.4/API/DES Cryptographic key destruction

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]

FCS_CKM.4.1/API/DES The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method *zeroing*[21] that meets the following: *No Standard*[22].

Application Note: This SFR concerns the destruction of temporal key storage cryptographic operation.

6.1.1.9 FCS_COP.1/API/AES Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/AES The TSF shall perform *encryption/decryption*[23] in accordance with a specified cryptographic algorithm *AES ECB, AES CBC, AES CTR modes*[24] and cryptographic key sizes *128 bit, 192 bit, 256 bit*[25] that meet the following: *[FIPS PUB 197] chapter 5, [NIST SP 800-38A], PKCS#7*[26].

6.1.1.10 FCS_COP.1/API/TDES Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/TDES The TSF shall perform *encryption/decryption*[27] in accordance with a specified cryptographic algorithm *TDES ECB, DES CBC modes*[28] and cryptographic key sizes *112 bit, 168 bit*[29] that meet the following*: [FIPS 197], [NIST SP 800-38A], PKCS#7*[30].

[21] [assignment: *cryptographic key destruction method*]
[22] [assignment: *list of standards*]
[23] [assignment: *list of cryptographic operations*]
[24] [assignment: *cryptographic algorithm*]
[25] [assignment: *cryptographic key sizes*]
[26] [assignment: *list of standards*]
[27] [assignment: *list of cryptographic operations*]
[28] [assignment: *cryptographic algorithm*]
[29] [assignment: *cryptographic key sizes*]
[30] [assignment: *list of standards*]

6.1.1.11 FCS_COP.1/API/RSA Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/RSA The TSF shall perform *encryption/decryption/signature/verification*[31] in accordance with a specified cryptographic algorithm *RSA*[32] and cryptographic key sizes *from 2048-bit up to 4096-bit with 2-bit granularity*[33] that meet the following: *[ISO/IEC14888-2:2008] section 6.2 and 6.3, PKCS #1 v1.5 or PKCS PSS padding*[34].

6.1.1.12 FCS_COP.1/API/ECDH Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/ECDH The TSF shall perform *key agreement*[35] in accordance with a specified cryptographic algorithm *ECDH*[36] and cryptographic key sizes *from 256-bit up to 521-bit*[37] that meet the following: *[ANS X9.63], section 5.4.1 Standard Diffie-Hellman primitive*[38].

6.1.1.13 FCS_COP.1/API/SHA Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/SHA The TSF shall perform *hashing*[39] in accordance with a specified cryptographic algorithm *SHA256, SHA384 and SHA512*[40] and cryptographic key sizes *Not Applicable*[41] that meet the following: *[FIPS 180-4]*[42].

---

[31] [assignment: *list of cryptographic operations*]
[32] [assignment: *cryptographic algorithm*]
[33] [assignment: *cryptographic key sizes*]
[34] [assignment: *list of standards*]
[35] [assignment: *list of cryptographic operations*]
[36] [assignment: *cryptographic algorithm*]
[37] [assignment: *cryptographic key sizes*]
[38] [assignment: *list of standards*]
[39] [assignment: *list of cryptographic operations*]
[40] [assignment: *cryptographic algorithm*]
[41] [assignment: *cryptographic key sizes*]
[42] [assignment: *list of standards*]

6.1.1.14 FCS_COP.1/API/CMAC Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/CMAC The TSF shall perform *signature*[43] in accordance with a specified cryptographic algorithm *CMAC/AES*[44] and cryptographic key sizes *128, 192, 256 bits*[45] that meet the following: *[RFC 4493]*[46].

6.1.1.15 FCS_COP.1/API/GCM Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/GCM The TSF shall perform *encryption/decryption*[47] in accordance with a specified cryptographic algorithm *AES GCM mode*[48] and cryptographic key sizes *128, 192, 256 bits*[49] that meet the following: [*NIST SP 800-38D]*[50].

6.1.1.16 FCS_COP.1/API/HMAC Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/HMAC The TSF shall perform *keyed-Hash Message Authentication Code*[51] in accordance with a specified cryptographic algorithm *HMAC with sha-2*[52] and cryptographic key sizes *SHA-2-256/384/512*[53] that meet the following: *[FIPS PUB 198-1]*[54].

---

[43] [assignment: *list of cryptographic operations*]
[44] [assignment: *cryptographic algorithm*]
[45] [assignment: *cryptographic key sizes*]
[46] [assignment: *list of standards*]
[47] [assignment: *list of cryptographic operations*]
[48] [assignment: *cryptographic algorithm*]
[49] [assignment: *cryptographic key sizes*]
[50] [assignment: *list of standards*]
[51] [assignment: *list of cryptographic operations*]
[52] [assignment: *cryptographic algorithm*]
[53] [assignment: *cryptographic key sizes*]
[54] [assignment: *list of standards*]

6.1.1.17 FCS_COP.1/API/KBKDF Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/KBKDF The TSF shall perform *key derivation*[55] in accordance with a specified cryptographic algorithm with *KBKDF*[56] and cryptographic key sizes *HMAC-SHA2-256/384/512, CMAC-AES-128*[57] that meet the following: *[SP800-108]*[58].

6.1.1.18 FCS_COP.1/API/PBKDF2 Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/PBKDF2 The TSF shall perform *key derivation*[59] in accordance with a specified cryptographic algorithm with *PBKDF2*[60] and cryptographic key sizes *HMAC-SHA2-256/384/512*[61] that meet the following: *[RFC2898]*[62].

6.1.1.19 FCS_COP.1/API/HKDF Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/API/HKDF The TSF shall perform *key derivation*[63] in accordance with a specified cryptographic algorithm with *HKDF*[64] and cryptographic key sizes *HMAC-SHA2-256/384/512*[65] that meet the following: *[RFC5869]*[66].

---

[55] [assignment: *list of cryptographic operations*]
[56] [assignment: *cryptographic algorithm*]
[57] [assignment: *cryptographic key sizes*]
[58] [assignment: *list of standards*]
[59] [assignment: *list of cryptographic operations*]
[60] [assignment: *cryptographic algorithm*]
[61] [assignment: *cryptographic key sizes*]
[62] [assignment: *list of standards*]
[63] [assignment: *list of cryptographic operations*]
[64] [assignment: *cryptographic algorithm*]
[65] [assignment: *cryptographic key sizes*]
[66] [assignment: *list of standards*]

6.1.1.20 FDP_SDI.1/API Stored data integrity monitoring

Hierarchical to: No other components.

Dependencies: No dependencies.

FDP_SDI.1.1/API The TSF shall monitor user data stored in containers controlled by the TSF for *integrity errors*[67] on all objects, based on the following attributes: *CRC32/SHA-256/RSA2048*[68].

6.1.2 Related to application execution

6.1.2.1 FDP_ACF.1/FIREWALL Security attribute based access control

Hierarchical to: No other components.

Dependencies: FDP_ACC.1 Subset access control
FMT_MSA.3 Static attribute initialization

FDP_ACF.1.1/FIREWALL The TSF shall enforce the *Execution Context Separation Access Control*[69] to objects based on the *following list*:

Subject: Platform, User Application,
Object: Physical Resources,
Operation: Code Execution, Access to Data[70].

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

Platform is allowed to access any Physical Resource,
User Application is allowed to Execute the Code and Access to Data using the Physical Resource according to the permissions managed by Platform[71].

FDP_ACF.1.3/FIREWALL The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: *None*[72].

FDP_ACF.1.4/FIREWALL The TSF shall explicitly deny access of subjects to objects based on the following additional rules: *User Application is not allowed to Access to Data belonging to Platform or other User Application*[73].

[67] [assignment: *integrity errors*]
[68] [assignment: *user data attributes*]
[69] [assignment: *access control SFP*]
[70] [assignment: *list of subjects and objects controlled under the indicated SFP, and for each, the SFP-relevant security attributes, or named groups of SFP-relevant security attributes*]
[71] [assignment: *rules governing access among controlled subjects and controlled objects using controlled operations on controlled objects*]
[72] [assignment: *rules, based on security attributes, that explicitly authorise access of subjects to objects*]
[73] [assignment: *rules, based on security attributes, that explicitly deny access of subjects to objects*]

6.1.2.2 FDP_ACC.1/FIREWALL Subset access control

Hierarchical to: No other components.

Dependencies: FDP_ACF.1 Security attribute based access control

FDP_ACC.1.1/FIREWALL The TSF shall enforce the *Execution Context Separation Access Control*[74] on the *following list*:

Subject: Platform, User Application,
Object: Physical Resources,
Operation: Code Execution, Access to Data[75].

6.1.2.3 FMT_MSA.3/FIREWALL Static attribute initialisation

Hierarchical to: No other components.

Dependencies: FMT_MSA.1 Management of security attributes FMT_SMR.1 Security roles

FMT_MSA.3.1/FIREWALL The TSF shall enforce the *Execution Context Separation Access Control, access control policy for each user application by platform*[76] to provide *restrictive, none*[77] default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/FIREWALL The TSF shall allow the *Platform*[78] to specify alternative initial values to override the default values when an object or information is created.

6.1.2.4 FMT_MSA.1/FIREWALL Management of security attributes

Hierarchical to: No other components.

Dependencies: [FDP_ACC.1 Subset access control, or FDP_IFC.1 Subset information flow control] FMT_SMR.1 Security roles FMT_SMF.1 Specification of Management Functions

FMT_MSA.1.1/FIREWALL The TSF shall enforce the *Execution Context Separation Access Control, access control policy for each user application by platform*[79] to restrict the ability to *query, none*[80] the security attributes *each permission for read or write for memory*[81] to *each User Application*[82].

6.1.2.5 FMT_SMR.1/FIREWALL Security roles

Hierarchical to: No other components.

---

[74] [assignment: *access control SFP*]
[75] [assignment: *list of subjects, objects, and operations among subjects and objects covered by the SFP*]
[76] [assignment: *access control SFP, information flow control SFP*]
[77] [selection, choose one of: *restrictive, permissive,* [assignment: *other property*]]
[78] [assignment: *the authorised identified roles*]
[79] [assignment: *access control SFP(s), information flow control SFP(s)*]
[80] [selection: *change_default, query, modify, delete,* [assignment: *other operations*]]
[81] [assignment: *list of security attributes*]
[82] [assignment: *the authorised identified roles*]

Dependencies: FIA_UID.1 Timing of identification

FMT_SMR.1.1/FIREWALL The TSF shall maintain the roles *Platform, User Application*[83].

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

6.1.2.6 FMT_SMF.1/FIREWALL Specification of Management Functions

Hierarchical to: No other components.

Dependencies: No dependencies.

FMT_SMF.1.1/FIREWALL The TSF shall be capable of performing the following management functions*: Access to Data or Code Execution on Physical Resources*[84].

6.1.2.7 FCS_COP.1/BOOT Cryptographic operation

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/BOOT The TSF shall perform *RSA signature generation and RSA signature verification, hashing*[85] in accordance with a specified cryptographic algorithm *RSA, SHA-256*[86] and cryptographic key sizes *2048 bits*[87] that meet the following: *[ISO/IEC14888-2:2008], [FIPS 180-4]*[88].

6.1.2.8 FDP_SDI.1/BOOT Stored data integrity monitoring

Hierarchical to: No other components.

Dependencies: No dependencies.

FDP_SDI.1.1/BOOT The TSF shall monitor user data stored in containers controlled by the TSF for *integrity errors*[89] on all objects, based on the following attributes: *checksum*[90].

Application Note: "user data" and "objects" mean here "Code of User Application"

6.1.2.9 FCS_COP.1/APP_HANDLE Cryptographic operation

Hierarchical to: No other components.

---

[83] [assignment: *the authorised identified roles*]
[84] [assignment: *list of management functions to be provided by the TSF*]
[85] [assignment: *list of cryptographic operations*]
[86] [assignment: *cryptographic algorithm*]
[87] [assignment: *cryptographic key sizes*]
[88] [assignment: *list of standards*]
[89] [assignment: *integrity errors*]
[90] [assignment: *user data attributes*]

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation] FCS_CKM.4 Cryptographic key destruction

FCS_COP.1.1/APP_HANDLE The TSF shall perform *RSA signature verification and hashing*[91] in accordance with a specified cryptographic algorithm *RSA, SHA-256*[92] and cryptographic key sizes *2048 bits*[93] that meet the following: *[ISO/IEC14888-2:2008], [FIPS 180-4]*[94].

6.1.2.10 FDP_SDI.1/APP_HANDLE Stored data integrity monitoring

Hierarchical to: No other components.

Dependencies: No dependencies.

FDP_SDI.1.1/APP_HANDLE The TSF shall monitor user data stored in containers controlled by the TSF for *integrity errors*[95] on all objects, based on the following attributes: *checksum*[96].

Application Note: "user data" and "objects" mean here "Code of User Application"

6.1.3 SFR related to Card Content Management

6.1.3.1 FDP_SDI.1/CCM Stored data integrity monitoring

Hierarchical to: No other components.

Dependencies: No dependencies.

FDP_SDI.1.1/CCM The TSF shall monitor user data stored in containers controlled by the TSF for *integrity errors*[97] on all objects, based on the following attributes: *checksum*[98].

Application Note: "user data" and "objects" mean here "Code of User Application"

6.1.4 General security and physical attacks

6.1.4.1 FAU_ARP.1 Security alarms

Hierarchical to: No other components.

Dependencies: FAU_SAA.1 Potential violation analysis

FAU_ARP.1.1 The TSF shall take

[91] [assignment: *list of cryptographic operations*]
[92] [assignment: *cryptographic algorithm*]
[93] [assignment: *cryptographic key sizes*]
[94] [assignment: *list of standards*]
[95] [assignment: *integrity errors*]
[96] [assignment: *user data attributes*]
[97] [assignment: *integrity errors*]
[98] [assignment: *user data attributes*]

raise an exception,
mute,
block the card session,
block the card/Platform[99]

upon detection of a potential security violation.

6.1.4.2 FPT_FLS.1 Failure with preservation of secure state

Hierarchical to: No other components.

Dependencies: No dependencies.

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

fails from FPT_TST.1 SFR,
malfunction from unexpected external operating conditions,
detection of abnormal behavior or unexpected memory content[100].

6.1.4.3 FPT_ITT.1 Basic internal TSF data transfer protection

Hierarchical to: No other components.

Dependencies: No dependencies.

FPT_ITT.1.1 The TSF shall protect TSF data from *disclosure*[101] when it is transmitted between separate parts of the TOE.

6.1.4.4 FPT_PHP.3 Resistance to physical attack

Hierarchical to: No other components.

Dependencies: No dependencies.

FPT_PHP.3.1 The TSF shall resist *physical disturbances*[102] to the *all TOE parts*[103] by responding automatically such that the SFRs are always enforced.

6.2 Security Functional Requirements Rationale

6.2.1 TOE security objectives coverage

The following table is dedicated to security objectives coverage by SFRs

[99] [assignment: *list of actions*]
[100] [assignment: *list of types of failures in the TSF*]
[101] [selection: *disclosure, modification*]
[102] [assignment: *physical tampering scenarios*]
[103] [assignment: *list of TSF devices/elements*]

	O.ID	O.FIREWALL	O.OPERATE	O.RESOURCES	O.ALARM	O.KEY_MNGT	O.CRYPTO	O.RNG
FCS_RNG.1/API			X					X
FCS_CKM.4/API/CMAC			X			X		
FCS_CKM.4/API/HMAC			X			X		
FCS_CKM.4/API/KDF			X			X		
FCS_CKM.4/API/RSA			X			X		
FCS_CKM.4/API/ECDH			X			X		
FCS_CKM.4/API/AES			X			X		
FCS_CKM.4/API/DES			X			X		
FCS_COP.1/API/DES			X				X	
FCS_COP.1/API/AES			X				X	
FCS_COP.1/API/RSA			X				X	
FCS_COP.1/API/ECDH			X				X	
FCS_COP.1/API/SHA			X				X	
FCS_COP.1/API/CMAC			X				X	
FCS_COP.1/API/GCM			X				X	
FCS_COP.1/API/HMAC			X				X	
FCS_COP.1/API/KBKDF			X				X	
FCS_COP.1/API/PBKDF2			X				X	
FCS_COP.1/API/HKDF			X				X	
FDP_SDI.1/API			X					
FDP_ACF.1/FIREWALL		X	X					
FDP_ACC.1/FIREWALL		X	X	X				
FMT_MSA.3/FIREWALL		X						
FMT_MSA.1/FIREWALL		X						

FMT_SMR.1/FIREWALL	X	X		X				
FMT_SMF.1/FIREWALL		X						
FCS_COP.1/BOOT			X				X	
FDP_SDI.1/BOOT			X					
FCS_COP.1/APP_HANDLE	X						X	
FDP_SDI.1/APP_HANDLE	X							
FDP_SDI.1/CCM								
FAU_ARP.1					X			
FPT_FLS.1					X			
FPT_ITT.1					X			
FPT_PHP.3					X			

6.2.2 TOE security objectives coverage –Rationale

The following section is dedicated to security objectives rationale.

O.ID is fulfilled by the following SFRs:

- FCS_COP.1/APP_HANDLE ensures ensure appropriate identification and authentication mechanisms.
- FDP_SDI.1/ APP_HANDLE ensures to check the integrity and identification of User Application.
- FMT_SMR.1/FIREWALL ensures to give permission after checking identification of User Applications.

O.FIREWALL is fulfilled by the following SFRs:

- FDP_ACF.1/FIREWALL and FDP_ACC.1/FIREWALL ensures the FIREWALL access control policy.
- FMT_MSA.3/FIREWALL and FMT_MSA.1/FIREWALL specify security attributes enabling to ensure the authenticity, integrity, and/or confidentiality of card management commands.
- FMT_SMR.1/FIREWALL and FMT_SMF.1/FIREWALL indirectly contribute to meet this objective.

O.OPERATE is fulfilled by the following SFRs:

- FCS_RNG.1/API ensures the cryptographic quality of random number generation.
- FCS_CKM.4/API/CMAC, FCS_CKM.4/API/HMAC, FCS_CKM.4/API/KDF, FCS_CKM.4/API/RSA, FCS_CKM.4/API/ECDH, FCS_CKM.4/API/AES and FCS_CKM.4/API/DES contributes in covering this security objective and controls the observation of the key destruction not to disclose the keys.

- FCS_COP.1/API/DES, FCS_COP.1/API/AES, FCS_COP.1/API/RSA, FCS_COP.1/API/ECDH, FCS_COP.1/API/SHA, FCS_COP.1/API/CMAC, FCS_COP.1/API/GCM, FCS_COP.1/API/HMAC contributes in covering this security objective and controls the observation of the cryptographic operations no to disclose the keys
- FCS_COP.1/API/KBKDF, FCS_COP.1/API/PBKDF2, FCS_COP.1/API/HKDF contributes in covering this security objective and controls the observation of the key derivation which may be used to disclose the keys.
- FDP_SDI.1/API ensures that integrity errors related to the sensitive API result are detected by the TOE.
- FDP_ACF.1/FIREWALL and FDP_ACC.1/FIREWALL is able to detect and block various failures or security violations during usual working.
- FCS_COP.1/BOOT and FDP_SDI.1/BOOT ensure verification process to be required to use the Kernel, Crypto, and Application functions.

O.RESOURCES is fulfilled by the following SFRs:

- FDP_ACC.1/FIREWALL controls Execution Context Separation Access Control about Platform and User Application.
- FMT_SMF.1/FIREWALL enforces the card management operations (Installation, etc.), the privileges by defining the protective actions for the corresponding commands.

O.ALARM is fulfilled by the following SFRs:

- FAU_ARP.1 defines TSF reaction upon detection of a potential security violation..
- FPT_FLS.1 requires the card to preserve a secure state by TSF when failures occur.
- FPT_ITT.1 ensures to protect TSF data from disclosure when it is transmitted between separate parts of the TOE.
- FPT_PHP.3 resists physical disturbances to the all TOE parts by responding automatically such that the SFRs are always enforced.

O.KEY_MNGT is fulfilled by the following SFRs:

- FCS_CKM.4/API/CMAC, FCS_CKM.4/API/HMAC, FCS_CKM.4/API/KDF, FCS_CKM.4/API/RSA, FCS_CKM.4/API/ECDH, FCS_CKM.4/API/AES and FCS_CKM.4/API/DES contributes in covering this security objective and controls the observation of the cryptographic operations which may be used to disclose the keys.

O.CRYPTO is fulfilled by the following SFRs:

- FCS_COP.1/API/DES, FCS_COP.1/API/AES, FCS_COP.1/API/RSA, FCS_COP.1/API/ECDH, FCS_COP.1/API/SHA, FCS_COP.1/API/CMAC, FCS_COP.1/API/GCM and FCS_COP.1/API/HMAC contributes in covering this security objective and controls the observation of the cryptographic operations no to disclose the keys.
- FCS_COP.1/API/KBKDF, FCS_COP.1/API/PBKDF2 and FCS_COP.1/API/HKDF contributes in covering this security objective and controls the observation of the key derivation which may be used to disclose the keys.

O.RNG is fulfilled by the following SFRs:

- FCS_RNG.1/API ensures the cryptographic quality of random number generation

6.2.3 Dependencies

SFR	Mandatory dependency according to [CC2]	Fulfilled by security requirements
FCS_RNG.1/API	No dependencies	No dependencies
FCS_CKM.4/API/CMAC	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_CKM.4/API/HMAC	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_CKM.4/API/KDF	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_CKM.4/API/RSA	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_CKM.4/API/ECDH	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_CKM.4/API/AES	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_CKM.4/API/DES	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_COP.1/API/DES	FCS_CKM.4/API/DES	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_COP.1/API/AES	FCS_CKM.4/API/AES	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_COP.1/API/RSA	FCS_CKM.4/API/RSA	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_COP.1/API/ECDH	FCS_CKM.4/API/ECDH	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_COP.1/API/SHA	FCS_CKM.4	No (see discussion below)
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_COP.1/API/CMAC	FCS_CKM.4/API/CMAC	Yes
	[FDP_ITC.1 or FDP_ITC.2 or	No (see discussion below)

	FCS_CKM.1]	
FCS_COP.1/API/GCM	FCS_CKM.4/API/AES	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_COP.1/API/HMAC	FCS_CKM.4/API/HMAC	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_COP.1/API/KBKDF	FCS_CKM.4/API/KDF	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_COP.1/API/PBKDF2	FCS_CKM.4/API/KDF	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FCS_COP.1/API/HKDF	FCS_CKM.4/API/KDF	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FDP_ACF.1/FIREWALL	FDP_ACC.1/FIREWALL and FMT_MSA.3/FIREWALL	Yes
FDP_ACC.1/FIREWALL	FDP_ACF.1/FIREWALL	Yes
FMT_MSA.3/FIREWALL	FMT_MSA.1/FIREWALL and FMT_SMR.1/FIREWALL	Yes
FMT_MSA.1/FIREWALL	[FDP_ACC.1/FIREWALL or FDP_IFC.1], FMT_SMR.1/FIREWALL and FMT_SMF.1/FIREWALL	Yes
FMT_SMR.1/FIREWALL	FIA_UID.1	No (see discussion below)
FMT_SMF.1/FIREWALL	No dependencies	No dependencies
FCS_COP.1/BOOT	FCS_CKM.4/API/RSA	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FDP_SDI.1/BOOT	No dependencies	No dependencies
FCS_COP.1/APP_HANDLE	FCS_CKM.4/API/RSA	Yes
	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	No (see discussion below)
FDP_SDI.1/APP_HANDLE	No dependencies	No dependencies

FAU_ARP.1	FAU_SAA.1	No (see discussion below)
FPT_FLS.1	No dependencies	No dependencies
FPT_ITT.1	No dependencies	No dependencies
FPT_PHP.3	No dependencies	No dependencies

Rationale for the exclusion of dependencies:

- The dependency FCS_CKM.1 of FCS_CKM.4/API/CMAC, FCS_CKM.4/API/HMAC, FCS_CKM.4/API/KDF, FCS_CKM.4/API/RSA, FCS_CKM.4/API/ECDH, FCS_CKM.4/API/AES, FCS_CKM.4/API/DES, FCS_COP.1/API/DES, FCS_COP.1/API/AES, FCS_COP.1/API/RSA, FCS_COP.1/API/ECDH, FCS_COP.1/API/CMAC, FCS_COP.1/API/GCM, FCS_COP.1/API/HMAC, FCS_COP.1/API/KBKDF, FCS_COP.1/API/PBKDF2, FCS_COP.1/API/HKDF is unsupported.
  The keys used for the cryptographic operations are not created by the TOE and not "imported" into the TOE: they are only provided by the API user.
- The dependency FCS_CKM.1 of FCS_COP.1/API/SHA is not necessary because SHA operation doesn't have a key.
- The dependency FIA_UID.1 is unsupported.
  The identification is covered by FCS_COP.1/APP_HANDLE and FDP_SDI.1/APP_HANDLE.
- The FCS_CKM.1 and, FCS_CKM.4 of FCS_COP.1/BOOT and FCS_COP.1/APP_HANDLE is unsupported.
  FCS_COP.1/BOOT and FCS_COP.1/APP_HANDLE just do the verification operation.
- The dependency FAU_SAA.1 of FAU_ARP.1 is unsupported
  The dependency of FAU_ARP.1 on FAU_SAA.1 assumes that a "potential security violation" generates an audit event.

6.3 Security Assurance Requirements Rationale

EAL5+ is required for this type of TOE and product since it is intended to defend against sophisticated attacks. This evaluation assurance level allows a developer to gain maximum assurance from positive security engineering based on good practices. In order to provide a meaningful level of assurance that the TOE provides an adequate level of defense against such attacks: the evaluators should have access to the low level design and source code. The lowest for which such access is required is EAL5.

Development security is concerned with physical, procedural, personnel and other technical measures that may be used in the development environment to protect the TOE and the embedding product. The standard ALC_DVS.1 requirement mandated by EAL5 is not enough. Due to the nature of the TOE, it is necessary to justify the sufficiency of these procedures to protect their confidentiality and integrity. ALC_DVS.2 has no dependencies.

The TOE is intended to operate in hostile environments. AVA_VAN.5 "Advanced methodical vulnerability analysis" is considered as the expected level for Java Card technology-based products hosting sensitive applications, in particular for payment and identity areas. AVA_VAN.5 has dependencies on ADV_ARC.1, ADV_TDS.4, ADV_IMP.1, AGD_PRE.1, and AGD_OPE.1. All of them are satisfied by EAL5.

6.3.1 Dependencies

Table 6-1 SARs Dependencies

Require-ments	CC Dependencies	Satisfied Dependencies
ADV_ARC.1	ADV_FSP.1, ADV_TDS.1	ADV_FSP.5, ADV_TDS.4
ADV_FSP.5	ADV_TDS.1, ADV_IMP.1	ADV_TDS.4, ADV_IMP.1
ADV_IMP.1	ADV_TDS.3, ALC_TAT.1	ADV_TDS.4, ALC_TAT.2
ADV_INT.2	ADV_IMP.1, ADV_TDS.3, 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	No Dependencies
ALC_CMC.4	ALC_CMS.1, ALC_DVS.1, ALC_LCD.1	ALC_CMS.5, ALC_DVS.2, ALC_LCD.1
ALC_CMS.5	No Dependencies	No Dependencies
ALC_DEL.1	No Dependencies	No Dependencies
ALC_DVS.2	No Dependencies	No Dependencies
ALC_LCD.1	No Dependencies	No Dependencies
ALC_TAT.2	ADV_IMP.1	ADV_IMP.1
ASE_CCL.1	ASE_ECD.1, ASE_INT.1, ASE_REQ.1	ASE_ECD.1, ASE_INT.1, ASE_REQ.2
ASE_ECD.1	No Dependencies	No Dependencies
ASE_INT.1	No Dependencies	No Dependencies
ASE_OBJ.2	ASE_SPD.1	ASE_SPD.1
ASE_REQ.2	ASE_ECD.1, ASE_OBJ.2	ASE_ECD.1, ASE_OBJ.2
ASE_SPD.1	No Dependencies	No Dependencies
ASE_TSS.1	ADV_FSP.1, ASE_INT.1, ASE_REQ.1	ADV_FSP.5, ASE_INT.1, ASE_REQ.2
ATE_COV.2	ADV_FSP.2, ATE_FUN.1	ADV_FSP.5, ATE_FUN.1
ATE_DPT.3	ADV_ARC.1, ADV_TDS.4, 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, AGD_OPE.1, AGD_PRE.1, ATE_COV.1, ATE_FUN.1	ADV_FSP.5, AGD_OPE.1, AGD_PRE.1, ATE_COV.2, ATE_FUN.1
AVA_VAN.5	ADV_ARC.1, ADV_FSP.4, ADV_TDS.3, ADV_IMP.1, AGD_OPE.1, AGD_PRE.1, ATE_DPT.1	ADV_ARC.1, ADV_FSP.5, ADV_TDS.4, ADV_IMP.1, AGD_OPE.1, AGD_PRE.1, ATE_DPT.3

7 STATEMENT OF COMPATIBILITY

The aim of this section is to demonstrate that this Security Target does not contradict the Security Target elements of the underlying platform [ICST].

7.1 Objectives from IC vs Objectives from this TOE

O from [ICST]	Categorization (relevant / not relevant) plus related O from this ST (if applicable)	Rationale
O.Leak-Inherent	Relevant O.OPERATE, O.ALARM	Used for internal protections against physical attacks
O.Phys-Probing		
O.Malfunction		
O.Phys-Manipulation		
O.Leak-Forced		
O.Abuse-Func		
O.Identification	Not relevant	Serial Number or Unique Identifier of IC
O.RND	Relevant O.RNG	Random Numbers for internal needs and/or externally visible services
O.Mem-Access	Relevant O.FIREWALL	Used for implementation of logical separation between User Applications and Platform
O.Cap_Avail_Loader	Not relevant	Concern Manufacturing steps, not applicable in operational step of this TOE
O.Ctrl_Auth_Loader	Not relevant	
O.TDES	Relevant O.CRYPTO	TDES for internal needs and/or externally visible services
O.AES	Relevant O.CRYPTO	AES for internal needs and/or externally visible services
O. Authentication	Not relevant	Concern Manufacturing steps, not applicable in operational step of this TOE
O.Prot_TSF_Confidentiality	Relevant O.FIREWALL	Fulfilled with logical separation between User Applications and Platform

7.2 OE from IC vs OE from this TOE

OE from [ICST]	Categorization (IrOE / CfPOE / SgOE)	Rationale
OE.Process-Sec-IC	IrOE	For developing and manufacturing phases of the platform – see ALC class
OE.Lim_Block_Loader	IrOE	
OE.Loader_Usage	IrOE	
OE.TOE_Auth	IrOE	
OE.Resp-Appl	IrOE	

7.3 SFRs from IC vs SFRs from this TOE

SFR from [ICST]	Categorization (IP_SFR / RP_SFR-SERV / RP_SFR-MECH)	Rationale
FRU_FLT.2	RP_SFR-MECH	IC SFRs used as a countermeasures to ensure a correct execution of embedded software
FPT_FLS.1	RP_SFR-MECH	
FMT_LIM.1	IP_SFR	Concern Manufacturing steps
FMT_LIM.2	IP_SFR	
FAU_SAS.1	IP_SFR	
FDP_SDC.1	RP_SFR-MECH	IC SFRs used as a countermeasures to ensure a correct execution of embedded software
FDP_SDI.2	RP_SFR-MECH	
FPT_PHP.3	RP_SFR-MECH	
FDP_ITT.1	RP_SFR-MECH	
FTP_ITT.1	RP_SFR-MECH	
FDP_IFC.1	RP_SFR-MECH	
FCS_RNG.1/PTG.2	RP_SFR-SERV	IC RNG mechanisms are used for the implementation of TOE RNG services
FCS_RNG.1/RGS-IC	RP_SFR-SERV	
FDP_ACC.1	RP_SFR-MECH	*Memory Access Control* from IC are used to implements logical separation between User Applications and Platform (/FIREWALL SFRs from this TOE)
FDP_ACF.1	RP_SFR-MECH	
FMT_MSA.3	RP_SFR-MECH	
FMT_MSA.1	RP_SFR-MECH	
FMT_SMF.1	RP_SFR-MECH	
FCS_COP.1/TDES	RP_SFR-SERV	IC cryptographic mechanisms are used for the implementation of TOE cryptographic services
FCS_CKM.4/TDES	RP_SFR-SERV	
FCS_COP.1/AES	RP_SFR-SERV	
FCS_CKM.4/AES	RP_SFR-SERV	

7.4 SAR from [ICST] vs SARs from this ST

SAR	Level in [ICST] (corresponding to EAL 6+)	Level in this TOE (EAL5+)
ASE_INT	1	1
ASE_CCL	1	1
ASE_SPD	1	1
ASE_ECD	1	1
ASE_OBJ	2	2
ASE_REQ	2	2
ASE_TSS	1	1
ADV_FSP	5	5
ADV_TDS	5	4
ADV_IMP	2	1
ADV_INT	3	2
ADV_ARC	1	1
ADV_SPM	1	-

AGD_PRE	1	1
AGD_OPE	1	1
ALC_CMC	5	4
ALC_CMS	5	5
ALC_DEL	1	1
ALC_DVS	2	2
ALC_LCD	1	1
ALC_TAT	3	2
ALC_FLR	-	-
ATE_FUN	2	1
ATE_COV	3	2
ATE_DPT	3	3
ATE_IND	2	2
AVA_VAN	5	5

8 TOE SUMMARY SPECIFICATION

This chapter provides the summary of the security functionalities of the TOE.

8.1 Card Content Management Functions

This section describes the security functionalities provided by Card Content Management. The Bootloader is booting securely.

8.1.1 System manager

The ArcaShield platform supports System manager. System handler and App handler in System manger are related with Card Content Management.
System handler supports functions about access control and integrity check.

Following SFR groups fall under this security function:

- FAU_ARP.1
- FDP_SDI.1/CCM

App handler describes how TOE handle initializing application and application's status and access right.

Following SFR groups fall under this security function:

- FCS_COP.1/APP_HANDLE
- FDP_SDI.1/APP_HANDLE

8.2 Execution Security Functions

This section describes the security functionalities provided by Execution.

8.2.1 Firewall Policy

The MPU protects the reliability of the TOE system by:

- Preventing user applications from corrupting data used by the operating system
- Allowing memory regions to be defined as read-only so that vital data can be protected
- Detecting unexpected memory accesses
- Applications have a dedicated data area

Following SFR's fall under this TSF:

- FDP_ACF.1/FIREWALL

- FDP_ACC.1/FIREWALL
- FMT_MSA.3/FIREWALL
- FMT_MSA.1/FIREWALL
- FMT_SMR.1/FIREWALL
- FMT_SMF.1/FIREWALL

8.2.2 Secure Boot

Before staring boot-up, Bootloader checks the integrity of firmware image to know whether it is attacked or not through SHA256.

After each booting, the first verification process is required to use the Kernel and Crypto functions.

Following SFR groups fall under this security function:

- FCS_COP.1/BOOT
- FDP_SDI.1/BOOT

8.3 API and Services Security Functions

This section describes the security functionalities provided by the API and services.

8.3.1 Crypto API

Through the Crypto API, it is possible to perform encryption and decryption using symmetric and asymmetric keys, as well as conduct signature/verification operations with various keys. Furthermore, key generation and creation using dtrng are also supported.

a. Random Number Generation

- FCS_RNG.1/API

b. Cryptographic operation

- FCS_COP.1/API/AES
- FCS_COP.1/API/DES
- FCS_COP.1/API/RSA
- FCS_COP.1/API/ECDH
- FCS_COP.1/API/SHA
- FCS_COP.1/API/CMAC
- FCS_COP.1/API/GCM
- FCS_COP.1/API/HMAC
- FCS_COP.1/API/KBKDF
- FCS_COP.1/API/PBKDF2

- FCS_COP.1/API/HKDF

8.3.2 Kernel API

Kernel API supports user application implementation. If customer wants to develop their application on the Arcashield platform, they can use these kernel API to use Arcashield platform's function. Kernel API supports for variable clock, random number and storage update.

- FPT_ITT.1
- FCS_CKM.4/API/CMAC
- FCS_CKM.4/API/HMAC
- FCS_CKM.4/API/KDF
- FCS_CKM.4/API/RSA
- FCS_CKM.4/API/ECDH
- FCS_CKM.4/API/AES
- FCS_CKM.4/API/DES

8.3.3 System API

System API is an API provided not to users, but for the use of system applications specific to the platform (e.g., FAULT). In the System API, there is a connection between the Target of Evaluation (TOE) and the APIs related to fault handling and firmware integrity checking.

- FAU_ARP.1
- FPT_FLS.1
- FPT_PHP.3
- FDP_SDI.1/API
- FPT_ITT.1
- FCS_CKM.4/API/CMAC
- FCS_CKM.4/API/HMAC
- FCS_CKM.4/API/KDF
- FCS_CKM.4/API/RSA
- FCS_CKM.4/API/ECDH
- FCS_CKM.4/API/AES
- FCS_CKM.4/API/DES

8.4 Secure HW Platform Security Functions

This section describes the security functionalities provided by the Secure HW Platform.

8.4.1 Platform Security

This security function ensures a secure state of information, the non-observability of operations on the information, and the unavailability of previous information content on de-allocation/allocation.

In case of abnormal event such as data unavailable on an allocation or illegal access to a data, the system owns an internal mechanism that stops the code execution and raises an exception.

Following are some of the security related features provided:

- Prevents deploying the Loader functionality
- Side Channel Attack Preventive measures
- All the security features mentioned in the section Platform support
- Fault detector counter

Following SFR's fall under this TSF:

- FAU_ARP.1
- FPT_FLS.1

8.4.2 Platform Support

The Hardware/IC (For more information, refer [ICST]) supports hardware countermeasures for SCA, Active Shield, Environmental & Life time detector & filters.

Following SFR's fall under this TSF:

- FPT_PHP.3
- FPT_ITT.1

9 ACRONYMS

Table 9-1 describes the acronyms used in this document.

Table 9-1 List of Acronyms

Acronym	Meaning
AAUI	Application Activation User Interface
ACE	Access Control Enforcer
ACP	Access Control Policy
ADELG	Applet Deletion Group
AES	Advanced Encryption Standard
AID	Application Identifier
AID	Applet IDentifier
AP	Application Provider
APDU	Application Protocol Data Unit
API	Application Programming Interface
APSD	Application Provider Security Domain
ARA	Access Rules Application
ARM	Advance RISC Machine
ATR	Answer To Reset
BIP	Bearer Independent Protocol
CA	Controlling Authority
CAD	Card Acceptance Device
CASD	Controlling Authority Security Domain
CBC	Cipher Block Chaining
CC	Common Criteria
CCM	Card Content Management
CCN	Centro Criptológico Nacional
CEM	Common Evaluation Methodology
CIN	Card Image Number / Card Identification Number
CLA	Instruction class (of an APDU Command)
CLF	Contactless Front End
CLFDB	Confidential Load File Data Block
CLT	Contactless Tunneling
CPLC	Card Production Life Cycle Data
CRC	Cyclic Redundancy Check
CVM	Cardholder Verification Method
DAP	Data Authentication Pattern
DEK	Data Encryption Key

DES	Data Encryption Standard
DFA	Differential Fault Analysis
DGI	Data Grouping Identifier
DPA	Differential Power Analysis
DV	Document Verifier
EAL	Evaluation Assurance Level
ECB	Electronic Code Book
ECDH	Elliptic Curve Diffie Hellman
EEPROM	Electrically Erasable Programmable Read Only Memory
EMA	Electro-Magnetic Analysis
EMV	Europay, MasterCard, and Visa; used to refer to the ICC Specifications for Payment Systems
ENC	Encryption
EPA	Emanation Power Analysis
eSE	Embedded Secure Element
ETR_COMP	Evaluation Test Report for a composite Smart Card Evaluation
ETSI	European Telecommunications Standards Institute
HAL	Hardware Abstraction Layer
HCI	Host Controller Interface
HW/SW/FW	Hardware/Software/Firmware
IC	Integrated Circuit
ICV	Initial Chaining Vector
IFSC	Information Field Size for Card
IFSD	Information Field Size for Interface Device
IIN	Issuer Identification Number
INS	Instruction code(of an APDU command)
ISCI	International Security Certification Initiative
ISD	Issuer Security Domain
ISO	International Organization for Standardization
ITSEF	Information Technology Security Evaluation Facility
Lc	Exact length of data in a case 3 or case 4 command
LCG	Logical Channel Group
Le	Maximum length of data expected in response to a case 2 or case 4 command
LFDB	Load File Data Block
LLC	Link Layer Control
LRC	Linear Redundancy Check
LV	Length Value
MAC	Message Authentication Code
MNO	Mobile Network Operator
NAD	Node Address
NFC	Near Field Communication
NOS	Native Operating System
OE	Operating Environment

OPEN	Open Platform Environment
OS	Operating System
OSP	Organizational Security Policy
P1	Reference control parameter 1
P2	Reference control parameter 2
PCB	Protocol Control Byte
PIN	Personal Identification Number
PKI	Public Key Infrastructure
PP	Protection Profile
PPSE	Proximity Payment System Environment
RMI	Remote Method Invocation
ROM	Read Only Memory
RSA	Rivest / Shamir / Adleman asymmetric algorithm
RTE	Run Time Environment
SAR	Security Assurance Requirement
SCP	Smart Card Platform
SCP02/03/11	Secure Channel Protocol 02/03/11
SD	Security Domain
SE	Secure Element
SFR	Security Functional Requirement
SIO	Serial IO
SPA	Simple Power Analysis
SPD	Security Problem Definition
SRS	Software Requirement Specification
SSD	Supplementary Security Domain
ST	Security Target
SW	Status Word
SWP	Single Wire Protocol
TLV	Tag Length Value
TOE	Target of Evaluation
TSF	TOE Security Functions
UART	Universal Asynchronous Receiver/ Transmitter
UICC	Universal Integrated Circuit Card

10 GLOSSARY

Table 10-1 describes the glossary for this documentation.

Table 10-1 List of Glossary

APDU	Application Protocol Data Unit is an ISO 7816-4 defined communication format between the eSE and the off-card applications.
APDU buffer	The APDU buffer is the buffer where the messages sent (received) by the card depart from (arrive to)..
Application Code Verification	A static analysis of an Executable Module to determine whether it respects the CAP format and satisfies some essential security properties, such as the absence of pointer arithmetic, uncontrolled control jumps, data-structure overflows, and so on
Application developer	Company, which develops secure or standard applications that is loaded onto the platform.
Application Provider (AP)	The entity or institution responsible for the applications and their associated services. An AP can be a financial institution (for example, a bank), a transport operator, or a third party service provider.
Application Session	The link between the Application and the external world during a Card Session starting with the Application selection and ending with the Application de-election or termination of the Card Session.
Asymmetric Cryptography	A cryptographic technique that uses two related transformations, a public transformation (defined by the Public Key component), and a private transformation (defined by the Private Key component). These two key components have a property such that it is computationally infeasible to discover the Private Key, even if the Public Key is known.
Certification body	State office, which manages the Common Criteria certification of the platforms and the secure applications.
Chip Manufacturer	The organization responsible for embedding the software of the Platform in the IC (“masking process”).
Controlling Authority (CA)	An entity independent from the one represented by the issuer on the eSE. This entity is responsible for securing the key creation process and personalizing the Application Provider Security Domain (APSD).
DPA	Differential Power Analysis is a form of side channel attack in which an attacker studies the power consumption of a cryptographic hardware device such as a smart card.
Embedded Software	Platform and Application
Firewall	Pre-issuance loaded software.
Host	The back end system that supports the smart card. Hosts perform functions such as authorization and authentication, card administration, download of post-issuance Application code and data and transactional processing

Initialization Data	Any data supplied by the Platform Developer that is injected into the non-volatile memory of the IC by the IC Manufacturer. These data are, for instance, used for initializing the platform, and to enforce the traceability and secure shipment between the phases.
Logical channel	A logical link to an application on the Ese.
Masking Process	The process of embedding the binary code of the Operating System, the Runtime Environment, the Issuer Security Domain, and a collection of applets into the IC chip.
Message Authentication Code (MAC)	A symmetric cryptographic transformation of data that provides the data origin authentication and data integrity.
NVRAM	Non-Volatile Random Access Memory, a type of memory that retains its contents when the power is turned off.
Object	An entity on which a Security Policy is enforced.
Platform Developer	The organization responsible for developing the code of the Platform software.
Private Key	The private component of an asymmetric key pair.
Public Key	The public component of an asymmetric key pair.
RAM	Random Access Memory, is a type of computer memory that can be accessed randomly
Retry Counter	A counter, used in conjunction with the Retry Limit, to determine when the attempts to present a CVM value shall be prohibited.
Retry Limit	The maximum number of times an invalid CVM value can be presented prior to the CVM handler prohibiting the further attempts to present a CVM value.
Secure Channel	A communication mechanism between an off card entity and a card that provides a level of assurance, to one or both entities.
Secure Channel Session	A session, during an Application Session, starting with the Secure Channel Initiation and ending with a Secure Channel Termination or termination of either the Application Session or Card Session.
Security Attribute	A logical entity used by a Security Policy to determine whether the outcome of a requested operation may succeed.
Security Domain	On-card entity providing support for the control, security, and communication requirements of the Application Provider.
Security Policy	A set of rules that regulate how certain assets are managed, protected, and/or distributed.
Session Key	A key whose lifetime is a card session.
Smart Card IC Provider	Company, which manages the development and the production of the IC.
Smart Card manufacturer	Company, which manages the manufacturing of the secure element, especially in the IC Card manufacturing phase.
Smart Card personalizer	Company, which personalizes the secure element.
Subject	An active entity within the TOE that causes information to flow among the objects or change the status of the system. It usually acts on the behalf of a user. Objects can be active and thus are also subjects of the TOE.
Supplementary Security Domain	Security Domain other than ISD.

Symmetric Cryptography	A cryptographic technique that uses the same secret key for the transformation of both the originator and the recipient.
User	Either an Application or an off-card entity (via an APDU command) that requests a Subject to perform some operations on an Object within the scope of a Security Policy.
Validation laboratory	Accredited Security laboratory approved by the Issuer, which manages the validation of the standard applications.
Verification Authority (VA)	A trusted third party that acts on behalf of the Issuer can be represented on the eSE. The major responsibility of the VA is to verify the application signatures (Mandated DAP) during the loading process.

11 REFERENCE DOCUMENTS

11.1 References

[CC1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, April 2017, Version 3.1 Revision 5, CCMB-2017-04-001.

[CC2] Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional requirements, April 2017, Version 3.1 Revision 5, CCMB-2017-04-002.

[CC3] Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance requirements, April 2017, Version 3.1 Revision 5, CCMB-2017-04-003.

[CEM] Common Methodology for Information Technology Security Evaluation, Evaluation methodology, April 2017, Version 3.1 Revision 5, CCMB-2017-04-004

[JIL-1] Composite product evaluation for Smart Cards and similar devices, Version 1.3, February 2015

[JIL-2] Guidance for smartcard evaluation, Version 2.0, February 2010

[ICPP] Security IC Platform Protection Profile, version 1.0, 15 June 2007

[ICST] Security Target Lite of S3SSE2A Version 0.1, 30th April 2024

[JCSPP] Java Card System – Closed Configuration Protection Profile v3.1, June 2020

[NIST SP 800-67] Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, revised January 2012, National Institute of Standards and Technology

[FIPS PUB 197] Federal Information Processing Standards (FIPS) Publication 197, Advanced Encryption Standard (AES), November 26, 2001

[NIST SP 800-38A] Recommendation for Block Cipher Modes of Operation, 2001, with Addendum Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode, October 2010

[ISO/IEC14888-2:2008] Information technology -- Security techniques -- Digital signatures with appendix -- Part 2: Integer factorization based mechanisms, 2008

[ANS X9.63] American National Standard X9.63-2001, Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, November 20, 2001

[FIPS 180-4] Federal Information Processing Standards Publication 180-4 SECURE HASH STANDARD U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, 2011 February, 11

[RFC 4493] The AES-CMAC Algorithm, June 2006

[NIST SP 800-38D] Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, November 2007

[FIPS PUB 198-1] The Keyed-Hash Message Authentication Code (HMAC), July 2008

[SP800-108] Recommendation for Key Derivation Using Pseudorandom Functions (Revised), October 2009

[RFC2898] PKCS #5: Password-Based Cryptography Specification Version 2.0, September 2000

[RFC5869] HMAC-based Extract-and-Expand Key Derivation Function (HKDF), May 2010

[JIL_COMP] Joint Interpretation Library - Composite product evaluation for Smart Cards and similar devices. Version 1.5.1. May 2018.

[ETRfc] Evaluation Technical Report (ETR for composition), S3SSE2A, V1.0, 30/04/2024

[ArcaShield_UM_OPE] Arcashield Platform OPE, V0.9, 25/04/2025

[ArcaShield_UM_PRE] Arcashield Platform PRE, V0.3, 10/05/2025