Security Target for Java Card Platform Implementation for Infineon on M7892 G12 (SLJ 52GxxyyyzC) V2.0

May 2019

ST Lite Version 3.6

Security Evaluations
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065

Security Target Lite for Java Card Platform Implementation for Infineon on M7892 G12 (SLJ 52GxxyyyzC) V2.0

Author: Oracle Corporation.

Contributors: Saqib Ahmad, Tyrone Stodart

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.
Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together

Table of Contents

CHAPTER

1 Introduction

This chapter provides the identification of the Security Target presents its general structure and introduces key notions used in the following chapters.

1.1 ST Reference

Title: Security Target Lite Java Card Platform Implementation for Infineon on M7892 G12 (SLJ 52GxxyyyzC) V2.0

Version: 3.6

Publication date: May 2019

Sponsor: Oracle Corporation, 500 Oracle Parkway, Redwood Shores, U.S.A.

Editor: Oracle Corporation, 500 Oracle Parkway, Redwood Shores, U.S.A.

1.2 TOE Reference

TOE name/version: Java Card Platform Implementation for Infineon on M7892 G12 (SLJ 52GxxyyyzC) V2.0

TOE Hardware ID: M7892 G12 (Smart Card IC)

TOE Platform Commercial ID's: SLE 78CLFX408APHM, SLE 78CLFX400VPHM with the cryptographic library RSA - EC - Toolbox

Hardware Certification ID: BSI-DSZ-CC-0891-V3-2018

The TOE name/id consists of the following placeholder:

- the first x is for the available interface (can be 'C', 'L', or 'D' for the Contact based, contact Less or Dual Interface)
- the second x is for the available cryptography (can be 'A' for symmetric and asymmetric cryptography, and 'B' for only symmetric cryptography)
- The number yyy the available user memory (can be one of the following sizes: 036, 064, 080, 128, 144, 160kB)
- the last letter z is a place holder for products that will be based on the TOE (can be 'A' for ePassport, 'B' for eDriving License, 'C' for Open Platform, or 'D' for National eID with applications)

1.3 TOE Overview

The TOE is a Java Card Platform (JCP) as illustrated in Figure 1 and is composed of a Smart Card Platform (SCP) and an embedded software (JCS[1]+ non-TSF parts[2]). The JCP is compliant with Java Card Specification (Classic Edition) v3.0.1 and GlobalPlatform Specification v.2.2. In particular, it implements the GlobalPlatform ID Configuration 1.0 [GP_ID].

The TOE allows post-issuance downloading of applications that have been previously verified by an off-card trusted IT component. The JCP is managed by Card Manager that is a part of the TOE. The Card Manager is a native application acting as applet without being it. The JCP is fully compliant with the Java Card Specification v3.0.1 excluding the optional part JCRMI which is not implemented by the TOE. There are no pre-issuance applets installed. Native code post-issuance downloading is out of scope.

The SCP is a certified hardware platform – the M7892 G12 (SLJ 52GxxyyyzC) with RSA2048 v2.07.003, EC v2.07.003, Toolbox v2.07.003 and Symmetric Crypto library v2.02.010 and with specific IC dedicated software (firmware).

The hardware platform is certified by the BSI under the certification ID: BSI-DSZ-CC-0891-V3-2018. The corresponding Security Target Lite is [ST_IC]. Not all functionality of the hardware is used by the "composite TOE"[3]. Therefore some parts of the IC platform are not part of the "composite TOE".

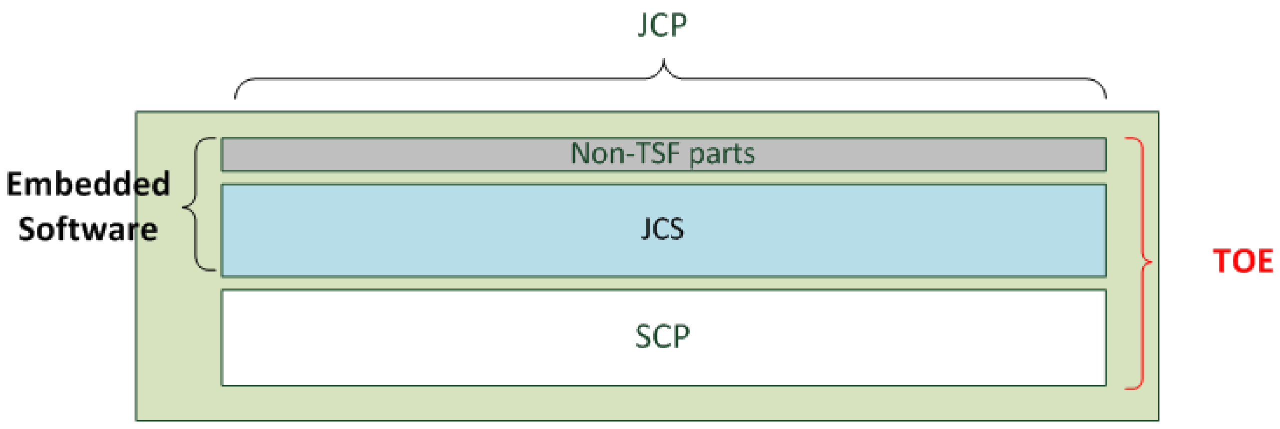


Figure 1 Java Card Platform overview

As the hardware platform offers a lot of flexibility regarding the size of the NVM, RAM size, co-processors, and software libraries, the TOE also supports a range of different options of the platform:

- Flash size ranges from minimal 0 kBytes to maximal 404 kBytes (in 1 kBytes Steps)

[1] The JCS covers the software TOE part developed by Oracle the Java Card RE, JC VM JC API, Card Manager, GP API, Native code) to be embedded on the SLE 78 chip.

[2] The non-TSF parts represent the modules which are out of the TOE Security Functionality scope but are part of the TOE.

[3] A "composite TOE" includes the SCP which has been already evaluated and the Embedded Software under evaluation.

- RAM for the user from minimal 0 kBytes to maximal 8 kBytes (in 1 kBytes Steps)
- ISO 14443 A/B Interface available or not
- The following cryptographic algorithms are also optional: RSA, ECC, and LDS (see chapter 1.4.1 for details).

Each of these options can be chosen by the order of the "composite TOE".

1.3.1 Usage and Major Security Features of the TOE

The TOE is a secure, generic platform that can be configured to a specific application by downloading one or more applets.

The JCP is compliant with Java Card Specification (Classic Edition) v.3.0.1 and GlobalPlatform Specification v.2.2 but not all functionality provided by JCP are included in the TOE. The table below lists major parts of the TOE dividing them into TSF and non-TSF parts.

Components		TSF parts	Non-TSF parts
SCP	Micro Controller	ISO 7816 Interface	Mifare-compatible interface
		ISO 14443 A/B Interface	
		Crypto2304T	
		SCP (AES and TDES)	
		TRNG	
	Crypto Library	RSA	Toolbox
		EC (prime and binary)	
		Symmetric Crypto Library	
	IC dedicated software	Firmware parts	
Embedded Software	Protocol	SCP02, SCP03	SCP01
	Cryptographic Algorithms	ECDSA (prime and binary)	Korean SEED MD5 RIPEMD160 MACs
		ECDH (prime and binary)	
		RSA	
		TDES	
		AES	
		RSA Key generation onboard	
		EC Key generation onboard	
		AIS20 DRG.4 (seeded from HW-TRNG)	
		SHA-1, SHA-2	
		Retail MAC, CMAC	
	Modules	LDS Supplementary Security Domains	Match-on-Card Biometric package Templating

Table 1: Major Security Features of the JCP

For a detailed description of the cryptographic algorithms implemented by the TOE please refer to definition of the FCS_COP elements in section 6.1.1.

1.3.2 TOE Type

The TOE is a Java Card Platform compliant with Java Card Specification (Classic Edition) v.3.0.1 and GlobalPlatform Specification v.2.2. The TOE allows post-issuance downloading of applications that have been previously verified by an off-card trusted IT component. It constitutes a secure generic platform that supports multi-application runtime environment and provides facilities for secure loading and interoperability between different applications. The TOE does not implement JCRMI and does not include any software on the application layer.

1.3.3 Required non-TOE Hardware/Software/Firmware

In the following the required non-TOE hardware/software/firmware is described on which the security of TOE relies.

1.3.3.1 The Bytecode Verifier

The bytecode verifier is a program that performs static checks on the bytecodes of the methods of a CAP file prior to the execution of the file on the card. Bytecode verification is a key component of security: applet isolation, for instance, depends on the file satisfying the properties a verifier checks to hold. A method of a CAP file that has been verified shall not contain, for instance, an instruction that allows forging a memory address or an instruction that makes improper use of a return address as if it were an object reference. In other words, bytecodes are verified to hold up to the intended use to which they are defined.

The TOE does not implement a bytecode verifier and fully relies on the off-card bytecode verification that has to be performed before a file is loaded on the card.

Different approaches have been proposed for the implementation of bytecode verifiers, most notably data flow analysis, model checking and lightweight bytecode verification, this latter being an instance of what is known as proof carrying code. The actual set of checks performed by the verifier is implementation dependent, but it is required that it should at least enforce all the “must clauses” imposed in chapter 7 of [JCVM3] on the bytecode and the correctness of the CAP files’ format.

1.4 TOE Description

The Java Card System (Java Card RE, Java Card VM, Java Card API, Card Manager, GlobalPlatform and the additional native code) is embedded in a Smart Card Platform. The Java Card RE, Java Card VM and Java Card API are compliant with Java Card specifications version 3, Classic Edition, including post-issuance downloading of applications verified off-card. All functionalities provided by the JCS are included in the TOE. The figure below shows the design of TOE outlining the TSF parts and non-TSF parts.

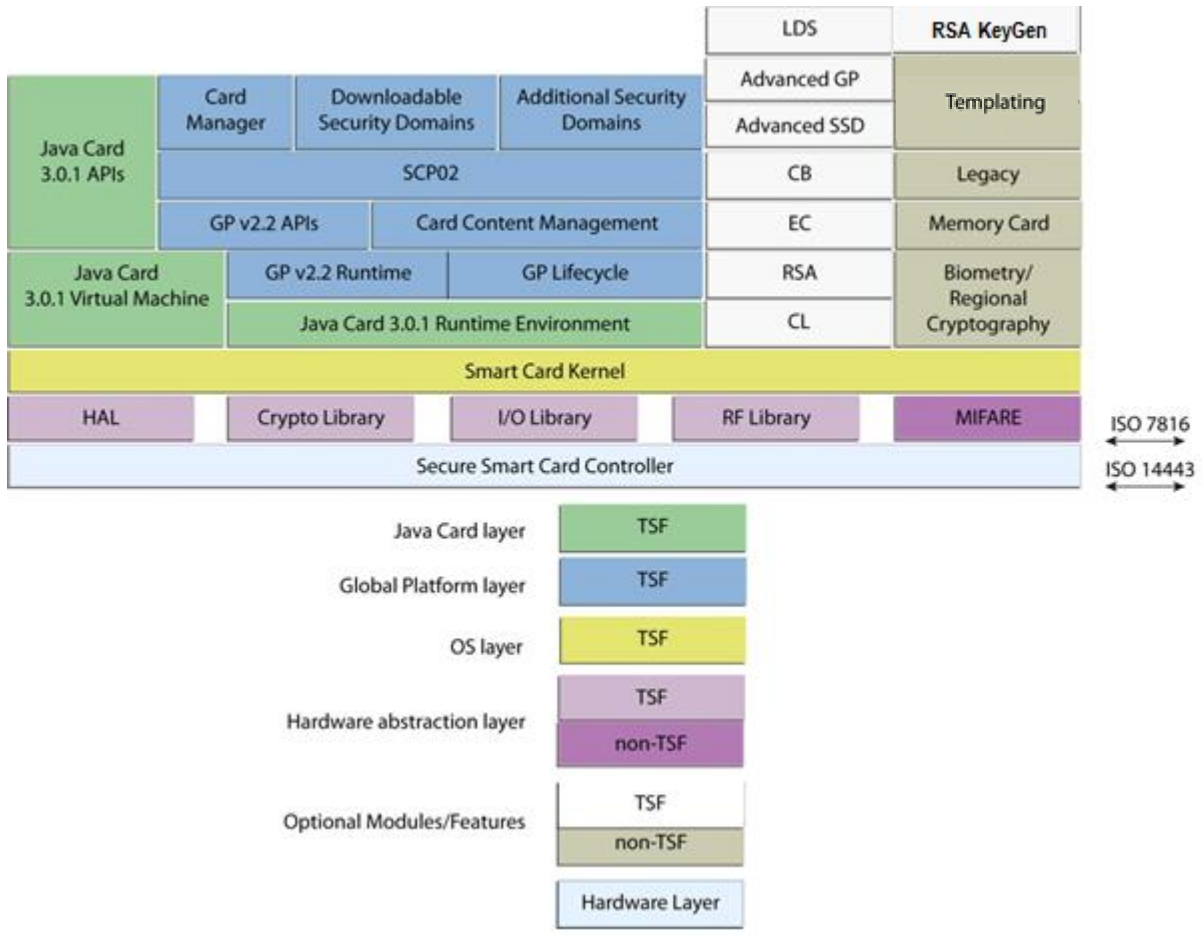


Figure 2 TOE components within the Java Card Platform

At the highest level, the components of JCP can be grouped as core and optional components, clearly marked in the figure above. Core components implement core functionality of the JCP and are required for all possible configurations. The core component group is made up of modules from the Java Card Platform, version 3.0.1 (green) the GlobalPlatform 2.2 (blue), and also the Secure Smart Card Controller (purple). In the figure's legend "**TSF**" indicates the component is part of the TOE Security Functionality.

Optional components implement optional functionality and are included in configurations as their functionality is required, see section 1.4.3 for details. The operating system and native primitives are tightly integrated with the Infineon libraries and hardware to provide maximum security and performance.

1.4.1 Java Card Platform components

In the following sections the core and optional components of JCP are described. The components that belong to the TOE Security Functionality are indicated with (TSF) and the components that are not part of the TOE Security Functionality are indicated with (non-TSF).

1.4.1.1 Core Components

Secure Smart Card Controller

The IC consists of a core system, memories, co-processors, peripherals, security modules and analogue peripherals. The major components of the core system are the two CPUs (Central Processing Units), the MMU (Memory Management Unit) and MED (Memory Encryption/Decryption Unit). The co-processor block contains the processors for RSA/EC and DES/AES processing, while the peripheral block contains the random number generation (True Random Number Generator) and the external interfaces service. This dual interface controller is able to communicate using either the contact based or the contactless interface.

Hardware Abstraction Layer

The implemented dual interface provides a maximum flexibility in using following communication protocols:

- ISO 7816 (**I/O Library**)(TSF),
- ISO 14443 Type A and Type B (**RF Library**) (TSF),
- ISO/IEC 18092 NFC passive mode (non-TSF)
- Mifare-compatible Interface (**MIFARE**) (non-TSF)

as well as further communication modes (non-TSF), allowing also the implementation of user defined concepts for contact based or contactless communication. The following proprietary extensions for the Infineon platform are also integrated as core components:

- Infineon Crypto API (non-TSF)

This package accelerates cryptographic functionality.

- Infineon Framework API (non-TSF)

This package provides a framework for efficiently accessing the TOE's chip functionality.

- (**Crypto Library**): this module covers the low-level cryptographic operations mostly using cryptographic hardware accelerators.
- (**HAL**): this module covers the low-level basic memory operations and other system level operation for the SLE78 chip. Provided service abstracts operations from underlying hardware service.

Smart Card Kernel

The Smart Card Kernel relies on the Smart Card Platform providing memory management, cryptography engine and input/output.

Java Card Layer,

Java Card technology combines a subset of the Java programming language with a runtime environment optimized for smart cards and related small-memory embedded devices. Java Card technology brings many of the benefits of the Java programming language to the resource-constrained world of smart cards.

The following are components of the Java Card Layer:

- **Java Card 3.0.1 APIs** (there are APIs that are not included in the TSF scope, that are the API which implement the functionality listed in the third column of Table 1)

The application programming interface for Java Card.

- **Java Card 3.0.1 Virtual Machine** (TSF)

The Java Card virtual machine is a subset of the Java virtual machine, and is designed to be run on smart cards and other resource-constrained devices. The Java Card virtual machine acts as an engine that loads Java class files and executes them with a particular set of semantics.

- **Java Card 3.0.1 Runtime Environment** (TSF)

A framework for running Java programs on the card.

<u>GlobalPlatform</u>

GlobalPlatform Card Specification 2.2 is an industry standard add-on layer that consists of a set of packages and APIs that standardize smart card functionality and provide a standardized infrastructure for the development, deployment and management of smart cards.

The following are components of the GlobalPlatform:

- **Card Manager** (TSF)

The card manager is an entity defined in the GlobalPlatform specification to enable the secure downloading of applications. The card manager implements the GlobalPlatform Environment (OPEN), the Issuer Security Domain and Cardholder verification Method Services.

- **Downloadable & Additional Security Domains** (TSF)

Provides an off-card entity the means for securely managing its applications. Security Domains are implemented by on-card privileged applets as specified in the GlobalPlatform specification and by the applet providers. The privileged applets that implement Security Domains use their keys and cryptographic material to provide a security "umbrella" under which the group of applications are managed. Cryptographic security provided by the security domains frees the applications from having to directly use Java Card platform functionality to implement security themselves.

The card supports multiple, dynamically configured Security Domains whose number is limited only by the available NVM. The card manager usually functions as a security domain called the Issuer Security Domain (ISD). The ISD is often the sole security domain, although the card can contain additional security domains called Supplementary Security Domains (SSDs).

Note that only the features enabling the Downloadable SDs are part of the TOE but not the SSDs themselves.

- **Secure Channels(SCP02)** (TSF)

A Secure channel is a communication mechanism between an off-card entity and a card that provides a level of assurance to one or both entities. The TOE supports Secure Channel Protocols SCP02 as defined in the GlobalPlatform specification.

- **GlobalPlatform API** (**GPv2.2 APIs**)(TSF)

The GlobalPlatform API provides services to applications, for example, cardholder verification, personalization, or security services. It also provides card content management services such as card locking and provides application life cycle state updates to applications.

- **Card Content Management** (TSF)

The card content management component governs loading, installation, extradition, registry update and removal of card content.

- **GlobalPlatform Runtime Environment(GP v2.2 Runtime)**(TSF)

The runtime environment provides an API for applications as well as a secure storage and execution space for applications that ensures that each application's code and data can remain separate and secure from other applications on the card. The card's runtime environment is also responsible for providing communication services between the card and off-card entities.

- **GlobalPlatform Life Cycle (GP lifecycle)** (TSF)

The life cycle component is responsible for maintaining the overall security and administration of the card and its contents throughout its life cycle.

1.4.1.2 Optional Components

The following components are available on the Java Card Platform Implementation for Infineon on M7892 G12 (SLJ 52GxxyyyzC) and can be included in configurations. The designation for each component on Figure 2 above is indicated at the end of its description below.

- **RSA** (TSF)

RSA is a public-key cryptography algorithm for which security is based on presumed difficulty of factoring of large integers. This component does not include RSA key generation. See RSA KeyGen.

- **RSA KeyGen** (TSF)

This component implements key generation for RSA. RSA is a public-key cryptography algorithm for which security is based on presumed difficulty of factoring of large integers.

- Elliptic Curve Cryptography (TSF)

Elliptic Curve Cryptography (**EC**): a public-key cryptography based on the algebraic structure of elliptic curves over finite fields.

- Contact Protocol(**CB**) (TSF)

This component implements ISO/IEC7816 specifications for smartcards.

- Contactless A/B Protocol(**CL**) (TSF)

This component implements ISO/IEC 14443 specifications for proximity contactless identification cards.

- LDS Secure Messaging Accelerator(**LDS**) (TSF)

This component implements secure messaging accelerator class for data transmission to and from a logical data store used for ePassports according to [TR03110] and [ICAO Doc 9303].

- **Advanced GP** (TSF)

Includes GP delegated management, GP extradition, and GP registry update.

- **Advanced SSD** (TSF)

Includes SCP03 and Downloadable SSD functionality.

- Korean SEED, MD5,RIPEMD-160, SCP01(**Legacy**) (non-TSF)

Korean Seed is a block cipher used broadly throughout South Korean industry. MD5 is the Message-Digest Algorithm, a cryptographic hash function that produces a 128-bit hash value used to check data integrity. RIPEMD-160 is RACE Integrity Primitives Evaluation Message Digest, a 160-bit message digest algorithm. SCP01 is a deprecated

GlobalPlatform secure channel protocol.

- Memory Card (non-TSF)

Provides memory card support for Mifare-compatibility.

- Match-on-Card (**Biometry/Regional Cryptography**) (non-TSF)

This component implements the extensible code area that can be populated with Match-on-Card technology that is used to match and store fingerprints on a smart card by extending Java Card and GlobalPlatform functionality.

- Templating (non-TSF)

This component enables reducing the production cost by effectively accelerating the complete pre-personalization step.

1.4.2 TOE Security Features

The TOE implements the Java Card Specifications versions 3 Classic Edition as specified by the Java Card Protection Profile - Open Configuration ([PP]). Since the TOE does not implement an on-card bytecode verifier, only application verified by an off-card verifier may be downloaded to the card. In the post-issuance stage where the applications may be loaded to the TOE in an adversary environment, the authenticity and the integrity of the files to be loaded are secured by means of mechanisms defined in [GP].

The next figure illustrates the process of integration of a new applet into TOE.

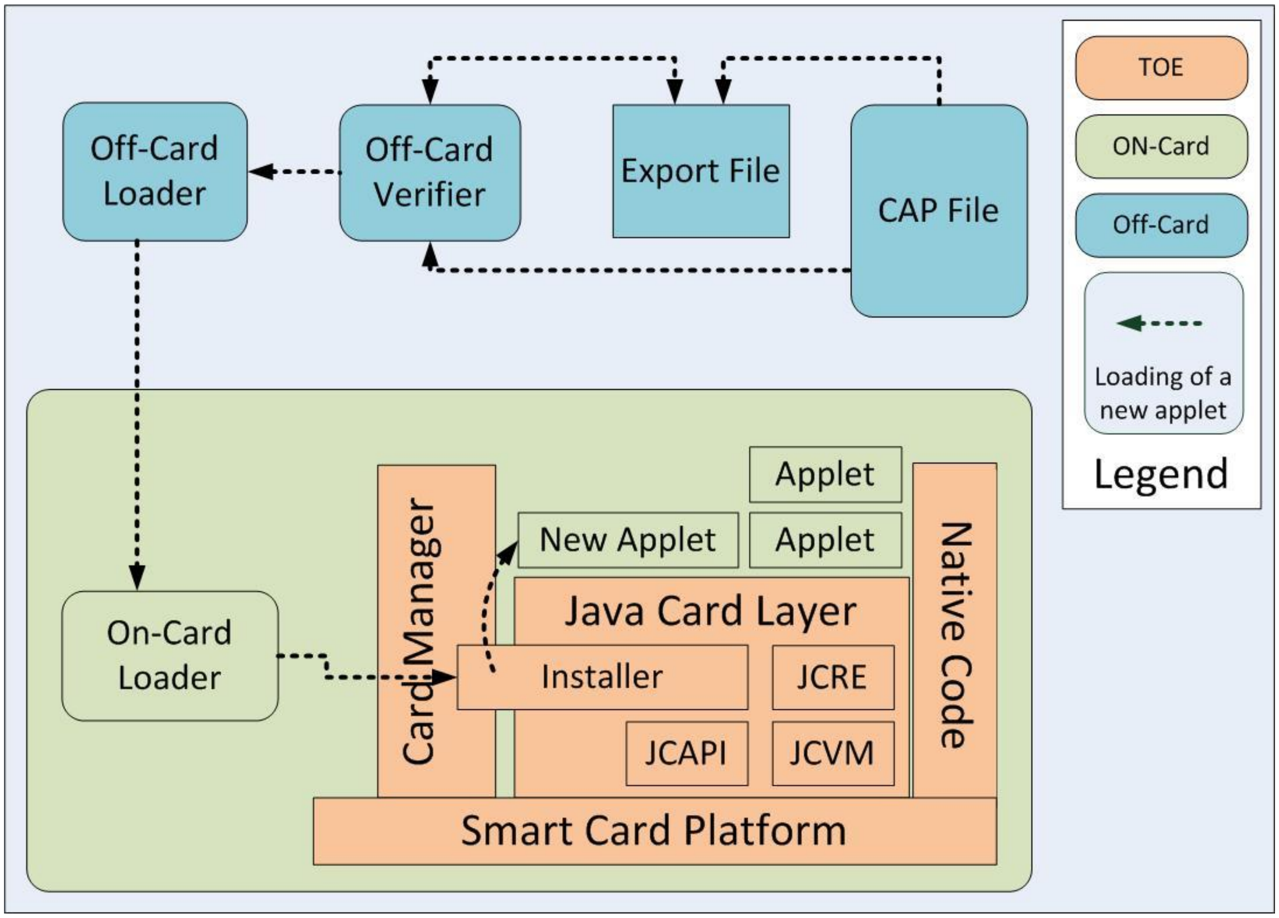


Figure 3 Java Card Platform and applet installation environment

The development of the applets is carried on in a Java programming environment. The compilation of the code produces the corresponding class file. Then, this latter file is processed by the converter[4] which validates the code and generates a converted applet (CAP) file, the equivalent of a Java class file for the Java Card platform. A CAP file contains an executable binary representation of the classes of a package. A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of Java Card technology, it defines either a user library, or one or several applets. Then, the off-card bytecode verifier checks the CAP file. After the validation is carried out, the CAP file has to be loaded into the card by means of a safe loading mechanism, where the off-card loader interacts with the on-card loader. The secure loading mechanism provided by the TOE is implemented according to GlobalPlatform specification. It provides means for preserving of authenticity and integrity of the files verified prior by the off-card verifier.

The loading of a file into the card embodies two main steps: First, an authentication step by which the card issuer and the card recognize each other as specified in [GP]. Once the identification step is accomplished, the CAP file is transmitted to the card. Due to resource limitations, usually the file is split by the card issuer into a list of Application Protocol Data Units (APDUs), which are in turn sent to the card. Each of these APDUs can be secured to preserve their

[4] The converter is defined in the specifications [JCVM3] as the off-card component of the Java Card virtual machine.

authenticity and integrity. Once loaded into the card the file is linked, what makes it possible in turn to install, if defined, instances of any of the applets defined in the file.

During the installation process the applet is registered on the card by using an application identifier (AID). This AID will allow the identification of unique applet instances within the card. In particular, the AID is used for selecting the applet instance for execution. In some cases, the actual installation (and registration) of applets is postponed; in the same vein, a package may contain several applets, and some of them might never be installed. Installation is then usually separated from the process of loading and linking a CAP file on the card.

The installer is the Java Card Layer component dealing with loading, linking and installation of new packages, as described in [JCRE3]. Once selected, it receives the CAP file, stores the classes of the package on the card, initializes static data, if any, and installs any applets contained in the package. The installer is also in charge of applet deletion as specified in [JCRE3], §11.3.4.

The Java Card VM is the bytecode interpreter as specified in [JCVM3]. The Java Card RE is responsible for card resource management, communication, applet execution, on-card system and applet security. The Java Card API provides classes and interfaces to the Java Card applets. It defines the calling conventions by which an applet may access the Java Card RE and native services such as, I/O management functions, PIN and cryptographic specific management and the exceptions mechanism.

While the Java Card VM is responsible for ensuring language-level security, the Java Card RE provides additional security features for Java Card technology-enabled devices. Applets from different vendors can coexist in a single card, and they can even share information. An apple, however, is usually intended to store highly sensitive information, so the sharing of that information must be carefully limited. In the Java Card platform, applet isolation is achieved through the applet firewall mechanism ([JCRE3] §6.1). That mechanism confines an applet to its own designated memory area, thus each applet is prevented from accessing fields and operations of objects owned by other applets, unless a "shareable interface" is explicitly provided (by the applet who owns it) for allowing access to that information. The Java Card RE allows sharing using the concept of "shareable interface objects" (SIO) and static public variables. Java Card VM dynamically enforces the firewall, that is, at runtime. However applet isolation cannot be entirely granted by the firewall mechanism if certain integrity conditions are not satisfied by the applications loaded on the card. Those conditions can be statically verified to hold by a bytecode verifier ([JCRE3], §6.1.1).

The Java Card VM ensures that the only way for applets to access any resources are either through the Java Card RE or through the Java Card API (or other vendor-specific APIs). This objective can only be guaranteed if applets are correctly typed (all the "must clauses" imposed in chapter 7 of [JCVM3] on the bytecodes and the correctness of the CAP file format are satisfied). The TOE does not implement an on-card verifier and fully relays on the off-card verifier. Each file must be verified by means of a trustful off-card verifier before loading into the card. The TOE provides security means according to [GP] with which the Issuer can preserve authenticity and integrity of the files to be loaded in an adversary environment.

Logical channels, as a part of the scope of the TOE, allow a terminal to open multiple sessions into the smart card, one session per logical channel ([JCRE3], §4). Commands may be issued on a logical channel to instruct the card either to open or to close a logical channel. An applet instance that is selected to be active on a channel shall process all the commands issued to that channel. The platform also introduces the possibility for an applet instance to be selected on multiple logical channels at the same time, or accepting other applets belonging to the same package to be selected simultaneously. These applets are referred to as multiselectable. A non-multiselectable applet can be active at most on one channel. Applets within a package are either all multiselectable or all non-multiselectable.

JCRMI is optional in version 3 Classic Edition and it is not included in this Security Target and out of the scope of this

TOE.

The TOE also includes the Card Manager (CM). The Card Manager is an application with specific rights, which is responsible for the administration of the smart card. It is conformant to the GlobalPlatform Card Specification 2.2 [GP] and is in charge of the life cycle of the whole card, as well as the installed applications (applets). The Card Manager supports multi-application runtime environment and provides via GlobalPlatform framework facilities for secure loading and interoperability between different applications. The card manager's role is also to manage and control the communication between the card and the card acceptance device (CAD) or the proximity-coupling device (PCD)[5] .

The card configuration can be set for GP mode or GP ID mode. If the card is configured for GP mode, SSDs follow the GlobalPlatform Card Specification v2.2 requirements [GP]. If the card is configured for GP ID mode, SSDs behave in accordance with the GlobalPlatform ID Configuration specification [GP_ID]. When GP ID mode is enabled, the ISD is required to use SCP 03 option 0x10, or if Advanced SSD is not present, SCP 02 option 0x55. SSDs, however, are allowed to use any of the following: SCP 03 option 0x00, SCP 03 option 0x10, or SCP 02 option 0x55. In this latter mode, the Card Manager supports the set of features defined in the GlobalPlaform Card ID Configuration. [GP_ID]. This latter document describes the mandatory and optional features operating in the GP ID mode of the TOE. The following defined optional features are nevertheless supported in the TOE: DAP Verification, Delegated Management, SCP02 option'55', SCP03 option '10', Supplementary Security Domains, Token Verification, Receipt generation.

Optionally, the TOE can be configured to behave like a static Java Card Platform where loading of applets is disabled.

The Java Card Layer also provides support for biometric templates management. Note that this feature is not part of the TSF.

1.4.3 Modularization

The JCP can be configured with a dedicated set of complete modules. Each of the modules can be chosen to be part of the TOE or not. The modules that belong to the TOE (TSF) and can be omitted are:

- RSA
- RSA KeyGen
- EC
- Contact Protocol (CB)
- Contactless Type A and/or Type B (CL)
- Advanced Supplementary Security Domain (Advanced SSD)
- LDS Secure Messaging Accelerator (LDS)
- Advanced GP

Please refer to Figure 2 for an overview of the TSF and non-TSF optional modules part of the TOE.

1.4.4 OS Extension Architecture

The TOE provides the ability to extend the JC/GP functionality by offering an extensible user code area (Sandbox) that

[5] The acronym CAD is used here and throughout this specification to refer to both types of card readers - the conventional Card Acceptance Device (CAD) for contacted I/O interfaces and the Proximity Coupling Device (PCD) for contactless interfaces

can be populated with custom code and be reachable from post-issuance Java applets via a secure, controlled mechanism. Note that the sandbox custom code is not part of the TOE. It is integrated into the final product, but provides no TSF.

[Confidential information removed in ST Lite]

The TOE's feature "OS extension architecture" covered by this Security Target is described in the [PP] as the "Extended Memory" feature, therefore the optional PP SFR package EMG is chosen (See Section 6.1.6).

1.4.5 Templating

When enabled by the chip manufacturer, templating provides a powerful feature intended to reduce the production cost by effectively accelerating the complete pre-personalization step.

[Confidential information removed in ST Lite]

1.5 Users and Roles

- **Platform Developer (Oracle, Phase 1):** Designs and implements the Embedded Software (JCS + non-TSF parts) to be embedded on the IC.
- **Chip/IC Manufacturer (Infineon, Phases 2-5):** Designs and implements the IC (SLE 78 chip) and prepares the packaging. The Chip Manufacturer prepares the smart card for platform initialization, modularizes the platform, modifies the Static Configuration parameters in the configuration table, merges the template image received by the Composite Product Integrator and may load the embedded software to the IC during wafer testing.
- **Composite Product Integrator (Infineon, Phase 5):** Pre-personalizes the smart card content by storing personalization data such as the SD keys or the public DAP verification key and install new Java Card Applets (pre-issuance). He can configure the TOE and do flashing. Configuration includes setting or changing the dynamic parameters of the card configuration table. He can run system tests to validate a template image candidate, load template or create a card template image (template dump) and send it back to the Chip Manufacturer for merging. The Composite Product Integrator integrates the Embedded Software within the IC (SLE78 chip) in order to produce a final smart card product ready for delivery to the Card Issuer. Note that Pre-issuance applet loading is enabled technically nevertheless no identified applets are to be loaded in pre-issuance in the scope of this evaluation. Otherwise, this will fall into the composite evaluation (Application upon Platform).
- **Card Issuer (Phases 6-7):** Issues smart cards to the Application Providers and the Cardholders. The Card Issuer controls the smart card's content and life cycle management during and after the platform being initialized, he personalizes the card by storing application user data.
- **Cardholder (Phase 7):** is the holder of the final smart card product (IC + Embedded Software + Application(s)).
- **Application Provider/Developer (Phases 6-7):** Develops Java Card applications providing services to the Cardholder requested by the Card Issuer. Application Providers are represented on the card by Supplementary Security Domains and depending on their privileges they can for instance download and install new Java Card Applets, modify the SSD keys, etc.
- **Verification Authority (Phases 6-7):** Is mainly responsible for the bytecode verification of the Java Card Applets loaded onto the TOE in post-issuance and checks that the Application Provider followed the operational

security guidance documentation.

1.6 TOE Life Cycle

The TOE life cycle, meaning in this process the IC with the entire software consisting of the operating system and the application(s), is part of the smart card life-cycle reaching from the first development steps to the final delivery to the end user.

The TOE life-cycle phases are illustrated in Figure 4 below. We refer to [PP0035] for a thorough description of Phases 1 to 7 of the Smart Card but the current scenario modifies the owner roles and places where the phases take place:

- Phases 1 and 2 compose the product development: Embedded Software (JCS + non-TSF parts development) and IC development.
- Phase 3 and Phase 4 correspond to IC manufacturing and packaging, respectively. Some IC initialization/pre-personalization steps may occur in Phase 3.
- Phase 5 concerns the embedding of software components within the IC. The Embedded Software flash image is securely stored, pre-personalized and tested on the TOE at the composite product integrator premises.
- Phase 6 is dedicated to the product personalization prior final use.
- Phase 7 is the product operational phase.

The Embedded Software (JCS + non-TSF parts) life cycle is composed of four stages:

- Embedded Software Development
- Embedded Software Storage, pre-personalization and testing
- Embedded Software Personalization
- Embedded Software Final usage.

Embedded Software storage is not necessarily a single step in the life cycle since it can be stored in parts. These stages map to the typical smartcard life cycle phases as shown in the following figure.

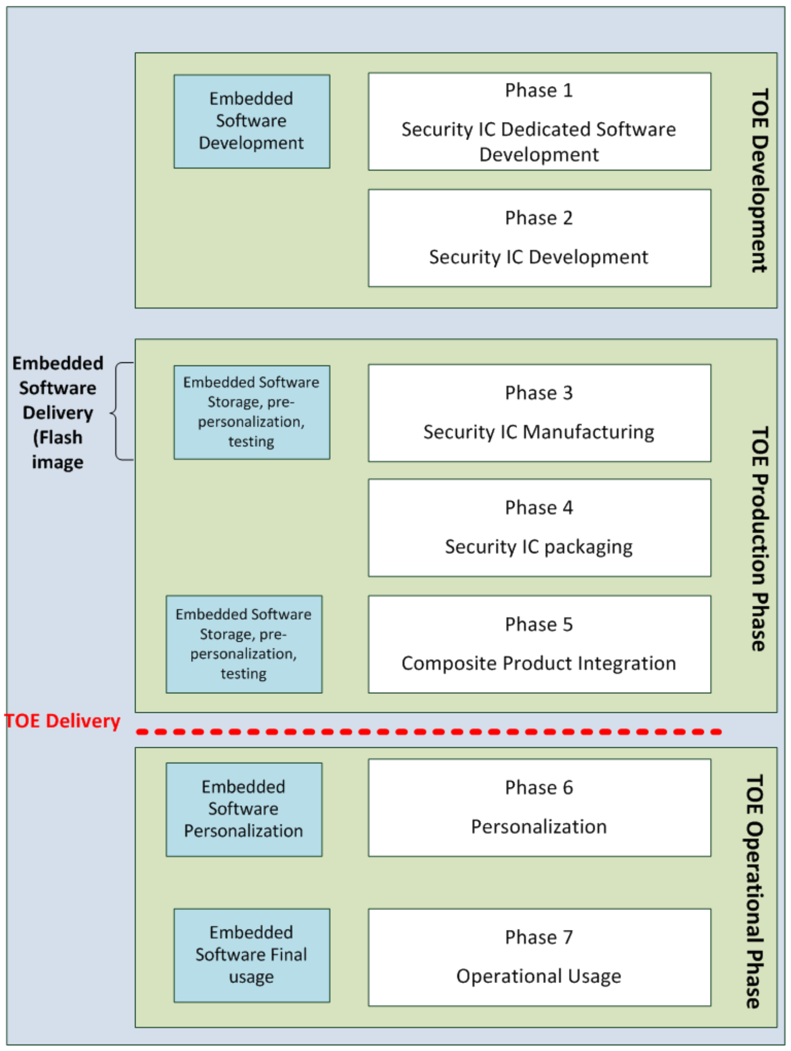

Embedded Software Development
Phase 1
Security IC Dedicated Software Development
Phase 2
Security IC Development
TOE Development
Embedded Software Delivery (Flash image
Embedded Software Storage, pre-personalization, testing
Phase 3
Security IC Manufacturing
Phase 4
Security IC packaging
Embedded Software Storage, pre-personalization, testing
Phase 5
Composite Product Integration
TOE Production Phase
TOE Delivery
Embedded Software Personalization
Phase 6
Personalization
Embedded Software Final usage
Phase 7
Operational Usage
TOE Operational Phase

Figure 4 TOE life Cycle within Smart Card Life Cycle

The Embedded Software (JCS + non-TSF parts) Development is performed during Phase 1 by the Platform Developer (Oracle). This includes Embedded Software conception, design, implementation, testing and documentation. The Embedded Software development fulfils requirements of the final TOE, including conformance to Java Card Specifications, and recommendations of the SCP user guidance. The Embedded Software development occurs in a secure audited environment maintaining the confidentiality of source code, data and any critical documentation and preserving the integrity of these elements. The evaluation of this product includes the Embedded Software development environment.

Phase 2 covers the Secure IC Development of Infineon and is out of scope of the TOE evaluation as that is part of the IC platform evaluation.

In Phase 3, the Chip Manufacturer (Infineon) stores and pre-personalizes the Embedded Software. The delivery from Oracle to Infineon of the Embedded Software (as a flash image) occurs at the beginning of Phase 3 (before the Chip Manufacturer performs his tasks). Delivery and acceptance procedures maintain the authenticity, the confidentiality and integrity of the exchanged delivery. The Embedded Software flash image delivery is done encrypted and signed. It requires therefore a previous secure exchange of public keys. The evaluation of this TOE includes the delivery process of the Embedded Software flash image to the Chip Manufacturer.

The Chip Manufacturer configures the TOE and applies the modularization (removal of optional code modules, integration of sandbox code, and configuration of the Embedded Software runtime switches) and can optionally merge the data template image received by the Composite Product Integrator ("template merge"). The Security IC Manufacturing environment protects the integrity and confidentiality of the Embedded Software and of any related material, for instance test suites. The evaluation of this TOE includes the whole Secure IC Manufacturing environment, in particular those locations where the Embedded Software is accessible for installation and testing. This certification process is based on the Common Criteria hardware certificate BSI-DSZ-CC-0782-2012-MA-01 and reuses the hardware evaluation testing results accordingly. At the end of Phase 3, the Chip Manufacturer

- delivers a flash image of the Embedded Software to the Composite Product Integrator through an encrypted APDU sequence (for "flashing")

or

- sends masked chips (IC with Embedded Software configured and/or merged, loaded via Wafer-Testing @ IFX.

Phase 4, is the IC packaging in Infineon and is part of the evaluation of the underlying platform. The process steps dealing with the hardware as such at the involved production sites were subject of hardware related site audits and are covered by the above mentioned hardware certificate. Nevertheless, the audit and evaluation of the templating processes at Infineon was not covered by the BSI-DSZ-CC-0782-MA01. During Phase 5, The Composite Product Integrator securely pre-personalizes the TOE (Key exchange, Security Domains management, Applet Loading). Then, the TOE is delivered to the Card Issuer for final personalization.

A Composite Product Integrator can perform one of the following options:

1. Pre-personalize the TOE by template loading on the configured Embedded Software.
2. Perform a template dump and use it for further production steps:
    - template merge done by the IC manufacturer or
    - template load done by the Composite Product Integrator).
3. If applicable, Pre-personalize the TOE by flashing[6] the encrypted flash image of the configured Embedded Software[7].

The pre-personalization steps in Phase 5 are done on the Composite Product Integrator's production sites. TOE pre-personalization covers the TOE configuration tuning (changes to the configuration table static/dynamic parameters), storing pre-personalization data (e.g. IFX card info) in the CPLC audit records following GP specs, storing Keys, etc. Note that pre-personalization could also happen in Phase 3 where Static Configuration Parameters could be changed whereas in phase 5, only Dynamic Configuration Parameters could be changed.

The delivery of the TOE (in the sense of the CC) to the Card Issuer is performed at the end of Phase 5 by the role Composite Product Integrator, i.e. Infineon.

[6] "Flashing" is storing code and data in the Solid Flash® NVM using the IFX Flash Loader Mechanism.

[7] An Embedded Software is considered configured once it has been initialized by the configuration data (via the configuration tool) and modularization was done.

Besides, the Flash Loader[8] could be used if needed during phase 5 but is finally locked before the TOE delivery to Phase 6.

Note that the complete phase 5 with all its production steps is performed by the role Composite Product Integrator and that for the scope of this evaluation this role is exclusively filled out by Infineon (and its specific template dump, load and merge production steps as well as flash loading).

The Embedded Software is personalized in Phase 6.

Note that this evaluation covers the Embedded Software development phase and all the TOE production phases where the Embedded Software is involved.

The Security Functional Requirements of the TOE are available only in the TOE Operational Phase of the smart card life cycle from the Personalization step in Phase 6 (once the card has been initialized) to the final operational usage in Phase 7. The preparation and personalization phases of the Embedded Software are covered by dedicated Preparation Guidance documents for the Composite Product Integrator and the IC Manufacturer.

1.7 Parts of the TOE

The following table lists the parts of the TOE that are delivered to the user.

Type	Description	Version
HW/SW	Java Card Platform Implementation for Infineon on M7892 G12 (SLJ 52GxxyyyzC)	V2.0
Documents	Operational User Guidance: • AGD_OPE • [Data_Book]	See Certification Report
Card manager Keyset	Transfer keyset for embedding	n/a
DAP Verification	Verification Authority's RSA public key	n/a

Note that the AGD_PRE Chip Manufacturer /Composite Product Integrator, the Configuration Tool[9]and the Templating Tools[10] are delivered to Infineon and used in the TOE production process.

[8] The Flash Loader is a firmware allowing the composite product integrator to download his code, or just parts of it, to the empty flash memory of the TOE.
[9] The configurator tool running in Dynamic Mode (used by the Composite Product Integrator) reads the dynamic properties in a configuration properties file and updates a card configuration by sending it configuration APDUs containing the dynamic properties. Please refer to the preparation guidance documents for more details. This tool is used by the IC Manufacturer or the Composite Product Integrator during Phase 5 of the life-cycle of the product. The configurator tool running in Static Mode reads the configuration properties file and outputs appropriately formatted configuration and hex files used by virtual cards and embedded cards, respectively
[10] Three production tools to achieve templating (dump tool, merge tool, and load tool (see Section 1.4.5 for more details)

1.8 TOE Usage

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 Java Card System is intended to transform a smart card into a platform capable of executing applications written in a subset of the Java programming language. The intended use of a Java Card platform is to provide a framework for implementing IC independent applications conceived to safely coexist and interact with other applications into a single smart card.

Applications installed on a Java Card platform can be selected for execution when the card communicates with a card reader.

Notice that these applications may contain other confidentiality (or integrity) sensitive data than usual cryptographic keys and PINs; for instance, passwords or pass-phrases are as confidential as the PIN, or the balance of an electronic purse.

Typical applications are:

- Financial applications, like Credit/Debit ones, stored value purse, or electronic commerce, among others.
- Transport and ticketing, granting pre-paid access to a transport system like the metro and bus lines of a city.
- Telephony, through the subscriber identification module (SIM) or the NFC chip for mobile phones.
- Personal identification, for granting access to secured sites or providing identification credentials to participants of an event.
- Electronic passports and identity cards.
- Secure information storage, like health records, or health insurance cards.
- Loyalty programs, like the “Frequent Flyer” points awarded by airlines. Points are added and deleted from the card memory in accordance with program rules. The total value of these points may be quite high and they must be protected against improper alteration in the same way that currency value is protected.

CHAPTER

2 Conformance Claims

2.1 CC Conformance

This Security Target is CC Part 2 [CC2] and CC Part 3 [CC3] conformant of Common Criteria version 3.1, revision 4.

Conformance of this ST is claimed for: Common Criteria part 2 extended by FPT_EMSEC.1, and FCS_RNG.1 and Common Criteria part 3 conformant.

All assurance components are taken from part 3. The Evaluation Assurance Level is EAL 5 augmented by AVA_VAN.5 "Advanced methodical vulnerability analysis" and ALC_DVS.2 "Sufficiency of security measures"

2.2 PP Conformance

This Security Target is in strict conformance to the Java Card Protection Profile Open Configuration Version 3.0 [PP].

2.3 PP Conformance Claim Rationale

The differences between this ST and the claimed Protection Profile are described below:

- This ST includes the SCP and the GlobalPlatform card manager to the composite product scope.
- The following security objectives for the operational environment have been moved to security objectives for the TOE in this ST: OE.CARD-MANAGEMENT (now: O.CARD-MANAGEMENT), OE.SCP.IC (now O.IC_SUPPORT), OE.SCP.RECOVERY (now O.RECOVERY), OE.SCP.SUPPORT (now O.OS_SUPPORT).
- The SFRs of the new group GPG were included to map the functionality of the SCP; the group SCPG contains the functionality to support the hardware platform.
- The following security objectives for the TOE were introduced: O.COMMUNICATION, O.EXT-MEM, and O.RND.
- The JCRMI functionality is not part of the TOE, so the related objective O.REMOTE is not included. This is intended by the PP.
- A.DELETION is not part of this ST, as the new objective O.DELETION is added.

CHAPTER

3 Security Aspects

This section is partly taken from [PP].

This chapter describes the main security issues of the Java Card System and its environment addressed in this Protection Profile, called “security aspects”, in a CC-independent way. In addition to this, they also give a semi-formal framework to express the CC security environment and objectives of the TOE. They can be instantiated as assumptions, threats, objectives (for the TOE and the environment) or organizational security policies.

The following sections present several security aspects from [PP] that are relevant for this ST.

3.1 Confidentiality

#.CONFID-APPLI-DATA Application data must be protected against unauthorized disclosure. This concerns logical attacks at runtime in order to gain read access to other application’s data.

#.CONFID-JCS-CODE Java Card System code must be protected against unauthorized disclosure. Knowledge of the Java Card System code may allow bypassing the TSF. This concerns logical attacks at runtime in order to gain a read access to executable code, typically by executing an application that tries to read the memory area where a piece of Java Card System code is stored.

#.CONFID-JCS-DATA Java Card System data must be protected against unauthorized disclosure. This concerns logical attacks at runtime in order to gain a read access to Java Card System data. Java Card System data includes the data managed by the Java Card RE, the Java Card VM and the internal data of Java Card platform API classes as well.

3.2 Integrity

#.INTEG-APPLI-CODE Application code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to the memory zone where executable code is stored. In post-issuance application loading, this threat also concerns the modification of application code in transit to the card.

#.INTEG-APPLI-DATA Application data must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain unauthorized write access to application data. In post-issuance application loading, this threat also concerns the modification of application data contained in a package in transit to the card. For instance, a package contains the values to be used for initializing the static fields of the package.

#.INTEG-JCS-CODE Java Card System code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to executable code.

#.INTEG-JCS-DATA Java Card System data must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to Java Card System data. Java Card System data includes the data managed by the Java Card RE, the Java Card VM and the internal data of Java Card API classes as well.

3.3 Unauthorized Executions

#.EXE-APPLI-CODE Application (byte)code must be protected against unauthorized execution. This concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language [JAVASPEC], §6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code; (3) unauthorized execution of a remote method from the CAD (if the TOE provides JCRMI functionality).

#.EXE-JCS-CODE Java Card System bytecode must be protected against unauthorized execution. Java Card System bytecode includes any code of the Java Card RE or API. This concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language ([JAVASPEC], §6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code. Note that execute access to native code of the Java Card System and applications is the concern of #.NATIVE.

#.FIREWALL The Firewall shall ensure controlled sharing of class instances11, and isolation of their data and code between packages (that is, controlled execution contexts) as well as between packages and the JCRE context. An applet shall not read, write, compare a piece of data belonging to an applet that is not in the same context, or execute one of the methods of an applet in another context without its authorization.

#.NATIVE Because the execution of native code is outside of the JCS TSF scope, it must be secured so as to not provide ways to bypass the TSFs of the JCS. Loading of native code, which is as well outside those TSFs, is submitted to the same requirements. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

3.4 Bytecode Verification

#.VERIFICATION Bytecode must be verified prior to being executed. Bytecode verification includes (1) how well-formed CAP file is and the verification of the typing constraints on the bytecode, (2) binary compatibility with installed CAP files and the assurance that the export files used to check the CAP file correspond to those that will be present on the card when loading occurs.

3.4.1 CAP File Verification

Bytecode verification includes checking at least the following properties: (3) bytecode instructions represent a legal set of instructions used on the Java Card platform; (4) adequacy of bytecode operands to bytecode semantics; (5) absence of operand stack overflow/underflow; (6) control flow confinement to the current method (that is, no control jumps to outside the method); (7) absence of illegal data conversion and reference forging; (8) enforcement of the private/public access modifiers for class and class members; (9) validity of any kind of reference used in the bytecodes (that is, any pointer to a bytecode, class, method, object, local variable, etc actually points to the beginning of piece of data of the expected kind); (10) enforcement of rules for binary compatibility (full details are given in [JCVM3], [JVM], [JCBV]). The actual set of checks performed by the verifier is implementation-dependent, but shall at least enforce all the “must clauses” imposed in [JCVM3] on the bytecodes and the correctness of the CAP files’ format.

[11] This concerns in particular the arrays, which are considered as instances of the Object class in the Java programming language.

As most of the actual Java Card VMs do not perform all the required checks at runtime, mainly because smart cards lack memory and CPU resources, CAP file verification prior to execution is mandatory. On the other hand, there is no requirement on the precise moment when the verification shall actually take place, as far as it can be ensured that the verified file is not modified thereafter. Therefore, the bytecodes can be verified either before the loading of the file on to the card or before the installation of the file in the card or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. This Protection Profile assumes bytecode verification is performed off-card.

Another important aspect to be considered about bytecode verification and application downloading is, first, the assurance that every package required by the loaded applet is indeed on the card, in a binary-compatible version (binary compatibility is explained in [JCVM3] §4.4), second, that the export files used to check and link the loaded applet have the corresponding correct counterpart on the card.

3.4.2 Integrity and Authentication

Verification off-card is useless if the application package is modified afterwards. The usage of cryptographic certifications coupled with the verifier in a secure module is a simple means to prevent any attempt of modification between package verification and package installation. Once a verification authority has verified the package, it signs it and sends it to the card. Prior to the installation of the package, the card verifies the signature of the package, which authenticates the fact that it has been successfully verified. In addition to this, a secured communication channel is used to communicate it to the card, ensuring that no modification has been performed on it.

3.4.3 Linking and Verification

Beyond functional issues, the installer ensures at least a property that matters for security: the loading order shall guarantee that each newly loaded package references only packages that have been already loaded on the card. The linker can ensure this property because the Java Card platform does not support dynamic downloading of classes.

3.5 Card Management

#.CARD-MANAGEMENT (1) The card manager (CM) shall control the access to card management functions such as the installation, update or deletion of applets. (2) The card manager shall implement the card issuer's policy on the card.

#.INSTALL (1) The TOE must be able to return to a safe and consistent state when the installation of a package or an applet fails or be cancelled (whatever the reasons). (2) Installing an applet must have no effect on the code and data of already installed applets. The installation procedure should not be used to bypass the TSFs. In short, it is an atomic operation, free of harmful effects on the state of the other applets. (3) The procedure of loading and installing a package shall ensure its integrity and authenticity.

#.SID (1) Users and subjects of the TOE must be identified. (2) The identity of sensitive users and subjects associated with administrative and privileged roles must be particularly protected; this concerns the JCRE, the applets registered on the card, and especially the default applet and the currently selected applet (and all other active applets in Java Card System). A change of identity, especially standing for an administrative role (like an applet impersonating the JCRE), is a severe violation of the TOE Security Policy (TSP). Selection controls the access to any data exchange between the TOE and the CAD and therefore, must be protected as well. The loading of a package or any exchange of data through the APDU buffer (which can be accessed by any applet) can lead to disclosure of keys, application code or data, and so on.

#OBJ-DELETION (1) De-allocation of objects should not introduce security holes in the form of references pointing to memory zones that are not longer in use, or have been reused for other purposes. Deletion of collection of objects should not be maliciously used to circumvent the TSFs.

(2) Erasure, if deemed successful, shall ensure that the deleted class instance is no longer accessible.

#DELETION (1) Deletion of installed applets (or packages) should not introduce security holes in the form of broken references to garbage collected code or data, nor should they alter integrity or confidentiality of remaining applets. The deletion procedure should not be maliciously used to bypass the TSFs. (2) Erasure, if deemed successful, shall ensure that any data owned by the deleted applet is no longer accessible (shared objects shall either prevent deletion or be made inaccessible). A deleted applet cannot be selected or receive APDU commands. Package deletion shall make the code of the package no longer available for execution. (3) Power failure or other failures during the process shall be taken into account in the implementation so as to preserve the SFRs. This does not mandate, however, the process to be atomic. For instance, an interrupted deletion may result in the loss of user data, as long as it does not violate the SFRs.
The deletion procedure and its characteristics (whether deletion is either physical or logical, what happens if the deleted application was the default applet, the order to be observed on the deletion steps) are implementation-dependent. The only commitment is that deletion shall not jeopardize the TOE (or its assets) in case of failure (such as power shortage).
Deletion of a single applet instance and deletion of a whole package are functionally different operations and may obey different security rules. For instance, specific packages can be declared to be undeletable (for instance, the Java Card API packages), or the dependency between installed packages may forbid the deletion (like a package using super classes or super interfaces declared in another package).

3.6 Services

#.ALARM The TOE shall provide appropriate feedback upon detection of a potential security violation. This particularly concerns the type errors detected by the bytecode verifier, the security exceptions thrown by the JCVM, or any other security-related event occurring during the execution of a TSF.

#.OPERATE (1) The TOE must ensure continued correct operation of its security functions. (2) In case of failure during its operation, the TOE must also return to a well-defined valid state before the next service request.

#.RESOURCES The TOE controls the availability of resources for the applications and enforces quotas and limitations in order to prevent unauthorized denial of service or malfunction of the TSFs. This concerns both execution (dynamic memory allocation) and installation (static memory allocation) of applications and packages.

#.CIPHER The TOE shall provide a means to the applications for ciphering sensitive data, for instance, through a programming interface to low-level, highly secure cryptographic services. In particular, those services must support cryptographic algorithms consistent with cryptographic usage policies and standards.

#.KEY-MNGT The TOE shall provide a means to securely manage cryptographic keys. This includes: (1) Keys shall be generated in accordance with specified cryptographic key generation algorithms and specified cryptographic key sizes, (2) Keys must be distributed in accordance with specified cryptographic key distribution methods, (3) Keys must be initialized before being used, (4) Keys shall be destroyed in accordance with specified cryptographic key destruction methods.

#.PIN-MNGT The TOE shall provide a means to securely manage PIN objects. This includes: (1) Atomic update of PIN value and try counter, (2) No rollback on the PIN-checking function,(3) Keeping the PIN value (once initialized) secret (for instance, no clear-PIN-reading function), (4) Enhanced protection of PIN's security attributes (state, try counter…) in confidentiality and integrity.

#.SCP The smart card platform must be secure with respect to the TSP. Then: (1) After a power loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state. (2) It does not allow the TSFs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System. (3) It provides secure low-level cryptographic processing to the Java Card System. (4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism. (5) It allows the Java Card System to store data in “persistent technology memory” or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low–level control accesses (segmentation fault detection). (6) It safely transmits low–level exceptions to the TOE (arithmetic exceptions, checksum errors), when applicable. We finally require that (7) the IC is designed in accordance with a well-defined set of policies and standards (likely specified in another protection profile), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

Note: For this TOE a certified hardware platform is used (see chapter 2).

#.TRANSACTION The TOE must provide a means to execute a set of operations atomically. This mechanism must not endanger the execution of the user applications. The transaction status at the beginning of an applet session must be closed (no pending updates).

CHAPTER

4 Security Problem Definition

This chapter describes the security problem to be addressed by the TOE and the operational environment of the TOE. The description is based on [PP] additional details of [PP0035].

4.1 Assets

Assets are security-relevant elements to be directly protected by the TOE. Confidentiality of assets is always intended with respect to un-trusted people or software, as various parties are involved during the first stages of the smart card product life-cycle; details are given in threats hereafter.

Assets may overlap, in the sense that distinct assets may refer (partially or wholly) to the same piece of information or data. For example, a piece of software may be either a piece of source code (one asset) or a piece of compiled code (another asset), and may exist in various formats at different stages of its development (digital supports, printed paper). This separation is motivated by the fact that a threat may concern one form at one stage, but be meaningless for another form at another stage.

The assets to be protected by the TOE are listed below. They are grouped according to whether it is data created by and for the user (User data) or data created by and for the TOE (TSF data). For each asset it is specified the kind of dangers that weigh on it.

4.1.1 User Data

D.APP_CODE The code of the applets and libraries loaded on the card. To be protected from unauthorized modification.
Note: This asset includes the code of the GlobalPlatform Framework on the card.

D.APP_C_DATA Confidential sensitive data of the applications, like the data contained in an object, a static field of a package, a local variable of the currently executed method, or a position of the operand stack.
To be protected from unauthorized disclosure.

D.APP_I_DATA Integrity sensitive data of the applications, like the data contained in an object, a static field of a package, a local variable of the currently executed method, or a position of the operand stack.
To be protected from unauthorized modification.

D.APP_KEYs Cryptographic keys owned by the applets.
To be protected from unauthorized disclosure and modification.

D.PIN Any end-user's PIN.
To be protected from unauthorized disclosure and modification.

D.CM_APDUs The APDU commands addressed to the Issuer Security Domain triggering a card management service. .
To be protected from unauthorized modification. In the case where an APDU contains a confidential asset such as the CM_KEYS value, this asset must be protected from unauthorized disclosure.

4.1.2 TSF Data

D.API_DATA Private data of the API, like the contents of its private fields.
To be protected from unauthorized disclosure and modification

D.CRYPTO Cryptographic data used in runtime cryptographic computations, like a seed used to generate a key.
To be protected from unauthorized disclosure and modification.

D.JCS_CODE The code of the Java Card System.
To be protected from unauthorized disclosure and modification

D.JCS_DATA The internal runtime data areas necessary for the execution of the Java Card VM, such as, for instance, the frame stack, the program counter, the class of an object, the length allocated for an array, any pointer used to chain data-structures.
To be protected from unauthorized disclosure or modification

D.SEC_DATA The runtime security data of the Java Card RE, like, for instance, the AIDs used to identify the installed applets, the currently selected applet, the current context of execution and the owner of each object.
To be protected from unauthorized disclosure and modification.

D.JCS_KEYS The cryptographic keys used when loading an executable file on the card.

To be protected from unauthorized disclosure and modification.

D.SEC_ATTRIBUTES The remaining security attributes related to the different security functionalities such as the number of remaining tries used during the cardholder authentication, the data contained in the GP registry (AID, privileges, lifecycle state, ...), etc.

To be protected from unauthorized modification

4.2 Threats

This section introduces the threats to the assets against which specific protection within the TOE or its environment is required. Several groups of threats are distinguished according to the configuration chosen for the TOE and the means used in the attack. The classification is also inspired by the components of the TOE that are supposed to counter each threat.

4.2.1 Confidentiality

T.CONFID-APPLI-DATA The attacker executes an application to disclose data belonging to another application. See #.CONFID-APPLI-DATA for details.
Directly threatened asset(s): D.APP_C_DATA, D.PIN and D.APP_KEYs.

T.CONFID-JCS-CODE The attacker executes an application to disclose the Java Card System code. See #.CONFID-JCS-CODE for details.
Directly threatened asset(s): D.JCS_CODE.

T.CONFID-JCS-DATA The attacker executes an application to disclose data belonging to the Java Card System. See #.CONFID-JCS-DATA for details.
Directly threatened asset(s): D.API_DATA, D.SEC_DATA, D.JCS_DATA, D.JCS_KEYS and D.CRYPTO.

4.2.2 Integrity

T.INTEG-APPLI-CODE The attacker executes an application to alter (part of) its own code or another application's code. See #.INTEG-APPLI-CODE for details.
Directly threatened asset(s): D.APP_CODE.

T.INTEG-APPLI-CODE.LOAD The attacker modifies (part of) its own or another application code when an application package is transmitted to the card for installation. See #.INTEG-APPLI-CODE for details.
Directly threatened asset(s): D.APP_CODE

T.INTEG-APPLI-DATA The attacker executes an application to alter (part of) another application's data. See #.INTEG-APPLI-DATA for details.
Directly threatened asset(s): D.APP_I_DATA, D.PIN and D.APP_KEYs.

T.INTEG-APPLI-DATA.LOAD The attacker modifies (part of) the initialization data contained in an application package when the package is transmitted to the card for installation. See #.INTEG-APPLI-DATA for details.
Directly threatened asset(s): D.CM_APDU, D.APP_I_DATA and D_APP_KEY.

T.INTEG-JCS-CODE The attacker executes an application to alter (part of) the Java Card System code. See #.INTEG-JCS-CODE for details.
Directly threatened asset(s): D.JCS_CODE.

T.INTEG-JCS-DATA The attacker executes an application to alter (part of) Java Card System or API data. See #.INTEG-JCS-DATA for details.
Directly threatened asset(s): D.API_DATA, D.SEC_DATA, D.SEC_ATTRIBUTES D.JCS_DATA and D.CRYPTO.

Other attacks are in general related to one of the above, and aimed at disclosing or modifying on-card information. Nevertheless, they vary greatly on the employed means and threatened assets, and are thus covered by quite different objectives in the sequel. That is why a more detailed list is given hereafter.

4.2.3 IDENTITY USURPATION

T.SID.1 An applet impersonates another application, or even the Java Card RE, in order to gain illegal access to some resources of the card or with respect to the end user or the terminal. See #.SID for details.
Directly threatened asset(s): D.SEC_DATA (other assets may be jeopardized should this attack succeed, for instance, if the identity of the JCRE is usurped), D.PIN D.APP_KEYs and D.JCS_KEYS

T.SID.2 The attacker modifies the TOE's attribution of a privileged role (e.g. default applet and currently selected applet), which allows illegal impersonation of this role. See #.SID for further details.
Directly threatened asset(s): D.SEC_DATA (any other asset may be jeopardized should this attack succeed, depending on whose identity was forged).

4.2.4 UNAUTHORIZED EXECUTION

T.EXE-CODE.1 An applet performs an unauthorized execution of a method. See #.EXE-JCS-CODE and #.EXE-

APPLI-CODE for details.
Directly threatened asset(s): D.APP_CODE.

T.EXE-CODE.2 An applet performs an execution of a method fragment or arbitrary data. See #.EXE-JCSCODE and #.EXE-APPLI-CODE for details.
Directly threatened asset(s): D.APP_CODE.

T.NATIVE An applet executes a native method to bypass a TOE Security Function such as the firewall. See #.NATIVE for details.
Directly threatened asset(s): D.JCS_DATA.

4.2.5 DENIAL OF SERVICE

T.RESOURCES An attacker prevents correct operation of the Java Card System through consumption of some resources of the card: RAM or NVRAM. See #.RESOURCES for details.
Directly threatened asset(s): D.JCS_DATA

4.2.6 CARD MANAGEMENT

T.DELETION The attacker deletes an applet or a package already in use on the card, or uses the deletion functions to pave the way for further attacks (putting the TOE in an insecure state). See #.DELETION for details).
Directly threatened asset(s): D.SEC_DATA and D.APP_CODE.

T.INSTALL The attacker fraudulently installs post-issuance of an applet on the card. This concerns either the installation of an unverified applet or an attempt to induce a malfunction in the TOE through the installation process. See #.INSTALL for details.
Directly threatened asset(s): D.SEC_DATA (any other asset may be jeopardized should this attack succeed, depending on the virulence of the installed application)

T.COMMUNICATION The attacker exploits the communication channel established between the TOE and CAD to modify or disclose confidential data.
Directly threatened asset(s): D.CM_APDUs , D.SEC_DATA and D.APP_CODE (DATA (any other asset may be threatened)

4.2.7 SERVICES

T.OBJ-DELETION The attacker keeps a reference to a garbage collected object in order to force the TOE to execute an unavailable method, to make it to crash, or to gain access to a memory containing data that is now being used by another application. See #.OBJ-DELETION for further details.
Directly threatened asset(s): D.APP_C_DATA, D.APP_I_DATA and D.APP_KEYs.

4.2.8 MISCELLANEOUS

T.PHYSICAL The attacker discloses or modifies the design of the TOE, its sensitive data or application code by physical (opposed to logical) tampering means. This threat includes IC failure analysis, electrical probing, unexpected tearing, and DPA. That also includes the modification of the runtime execution of Java Card System or SCP software through alteration of the intended execution order of (set of) instructions through physical tampering techniques. This threatens all the identified assets. This threat refers to the point (7) of the security aspect #.SCP, and all aspects related to confidentiality and integrity of code and data.

4.2.9 Threats Composite Platform

T.LEAKAGE An attacker may exploit information which is leaked from the TOE during usage of the Smart Card in order to disclose the confidential assets (TSF data or User data). No direct contact with the Smart Card Internals is required here. Leakage may occur through emanations, variations in power consumption, I/O characteristics, clock frequency, or by changes in processing time requirements. One example is the Differential Power Analysis (DPA).

T.FAULT An attacker may cause a malfunction of TSF by applying environmental stress in order to (1) deactivate or modify security features or functions of the TOE or (2) deactivate or modify security functions of the Smart Card. This may be achieved by operating the Smart Card outside the normal operating conditions.

The following threat is copied from [PP0035]

T.RND Deficiency of Random Numbers

An attacker may predict or obtain information about random numbers generated by the TOE for instance because of a lack of entropy of the random numbers provided.

An attacker may gather information about the produced random numbers which might be a problem because they may be used for instance to generate cryptographic keys.

4.3 Organisational Security Policies

This section describes the organizational security policies to be enforced with respect to the TOE environment.

OSP.VERIFICATION This policy shall ensure the consistency between the export files used in the verification and those used for installing the verified file. The policy must also ensure that no modification of the file is performed in between its verification and the signing by the verification authority. See #.VERIFICATION for details.
If the application development guidance provided by the platform developer contains recommendations related to the isolation property of the platform, this policy shall also ensure that the verification authority checks that these recommendations are applied in the application code.

4.4 Assumptions

This section introduces the assumptions made on the environment of the TOE.

A.APPLET Applets loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM3], §3.3) outside the API.

A.VERIFICATION All the bytecodes are verified at least once, before the loading, before the installation or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time.
This assumption is also upheld by the security objective of the environment OE.CODE-EVIDENCE which ensures that evidences exist that the application code has been verified and not changed after verification.

CHAPTER

5 Security Objectives

This section defines the security objectives to be achieved by the TOE.

5.1 TOE Security Objectives

5.1.1 IDENTIFICATION

O.SID The TOE shall uniquely identify every subject (applet, or package) before granting it access to any service.

5.1.2 EXECUTION

O.FIREWALL The TOE shall ensure controlled sharing of data containers owned by applets of different packages or the JCRE and between applets and the TSFs. See #.FIREWALL for details.

O.GLOBAL_ARRAYS_CONFID The TOE shall ensure that the APDU buffer that is shared by all applications is always cleaned upon applet selection. The TOE shall ensure that the global byte array used for the invocation of the install method of the selected applet is always cleaned after the return from the install method.

O.GLOBAL_ARRAYS_INTEG The TOE shall ensure that only the currently selected applications may have a write access to the APDU buffer and the global byte array used for the invocation of the install method of the selected applet.

O.NATIVE The only means that the Java Card VM shall provide for an application to execute native code is the invocation of a method of the Java Card API, or any additional API. See #.NATIVE for details.

O.OPERATE The TOE must ensure continued correct operation of its security functions. See #.OPERATE for details. O.REALLOCATION The TOE shall ensure that the re-allocation of a memory block for the runtime areas of the Java Card VM does not disclose any information that was previously stored in that block.

O.REALLOCATION The TOE shall ensure that the re-allocation of a memory block for the runtime areas of the Java Card VM does not disclose any information that was previously stored in that block.

Application note:

To be made unavailable means to be physically erased with a default value. Except for local variables that do not correspond to method parameters, the default values to be used are specified in

[JCVM3].

O.RESOURCES The TOE shall control the availability of resources for the applications. See #.RESOURCES for details.

5.1.3 SERVICES

O.ALARM The TOE shall provide appropriate feedback information upon detection of a potential security violation. See #.ALARM for details.

O.CIPHER The TOE shall provide a means to cipher sensitive data for applications in a secure way. In particular, the TOE must support cryptographic algorithms consistent with cryptographic usage policies and standards. See #.CIPHER for details.

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. See #.KEYMNGT.

O.PIN-MNGT The TOE shall provide a means to securely manage PIN objects. See #.PIN-MNGT for details.

Application note:

PIN objects may play key roles in the security architecture of client applications. The way they are stored and managed in the memory of the smart card must be carefully considered, and this applies to the whole object rather than the sole value of the PIN. For instance, the try counter's value is as sensitive as that of the PIN.

O.TRANSACTION The TOE must provide a means to execute a set of operations atomically. See #.TRANSACTION for details.

Application note: O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION and O.CIPHER are actually provided to applets in the form of Java Card APIs. Vendor-specific libraries can also be present on the card and made available to applets; those may be built on top of the Java Card API or independently. These proprietary libraries will be evaluated together with the TOE.

5.1.4 OBJECT DELETION

O.OBJ-DELETION The TOE shall ensure the object deletion shall not break references to objects. See #.OBJ-DELETION for further details.

5.1.5 APPLET MANAGEMENT

O.DELETION The TOE shall ensure that both applet and package deletion perform as expected. See #.DELETION for details.

O.LOAD The TOE shall ensure that the loading of a package into the card is safe. Besides, for code loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the verification authority. This verification by the TOE shall occur during the loading or later during the install process.
Application note: Usurpation of identity resulting from a malicious installation of an applet on the card may also be the result of perturbing the communication channel linking the CAD and the card. Even if the CAD is placed in a secure environment, the attacker may try to capture, duplicate, permute or modify the packages sent to the card. He may also try to send one of its own applications as if it came from the card issuer. Thus, this objective is intended to ensure the integrity and authenticity of loaded CAP files.

O.INSTALL The TOE shall ensure that the installation of an applet performs as expected (See #.INSTALL for details).
Besides, for code loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the verification authority. If not performed during the loading process, this verification by the TOE shall occur during the install process.

O.COMMUNICATION The TOE shall authenticate the origin of the card management requests that the card receives, and authenticate itself to the remote actor. It shall verify the integrity of the card management requests that it receives and be able, when it's needed, to process card management requests containing encrypted data.

O.RECOVERY If there is a loss of power, or if the smart card is withdrawn from the CAD while an operation is in progress, the SCP must allow the TOE to eventually complete the interrupted operation successfully, or recover to a consistent and secure state.

This security objective refers to the security aspect #.SCP(1): The smart card platform must be secure with respect to the SFRs. Then after a power loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state

O.CARD-MANAGEMENT The card manager shall control the access to card management functions such as the installation, update or deletion of applets. It shall also implement the card issuer's policy on the card.
The card manager is an application with specific rights, which is responsible for the administration of the smart card. This component will in practice be tightly connected with the TOE, which in turn shall very likely rely on the card manager for the effective enforcing of some of its security functions. Typically the card manager shall be in charge of the life cycle of the whole card, as well as that of the installed applications (applets). The card manager should prevent that card content management (loading, installation, deletion) is carried out, for instance, at invalid states of the card or by non-authorized actors. It shall also enforce security policies established by the card issuer.
Application note: The actor performing the card management operation must beforehand authenticate with the Security Domain. In the case of Delegated Management privilege, the card management command will be associated with an electronic signature (Verification token) verified by the ISD before execution.

O.IC_SUPPORT The SCP shall provide all IC security features against physical attacks. This security objective refers to the point (7) of the security aspect #.SCP:

It is required that the IC is designed in accordance with a well-defined set of policies and Standards (specified in another protection profile), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

O.OS_SUPPORT The SCP shall support the TSFs of the TOE. This security objective refers to the security aspects 2, 3, 4 and 5 of #.SCP:

(2) It does not allow the TSFs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System.

(3) It provides secure low-level cryptographic processing to the Java Card System.

(4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism.

(5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low-level control accesses (segmentation fault detection).

O.EXT-MEM The TOE shall provide controlled access means to the external memory and ensure that the external memory does not address Java Card System memory (containing User Data and TSF Data)

O.RND Random Numbers
The TOE will ensure the cryptographic quality of random number generation. For instance random numbers shall not be predictable and shall have sufficient entropy.
The TOE will 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

5.2 Security Objectives For The Operational Environment

This section introduces the security objectives to be achieved by the environment.

OE.APPLET No applet loaded post-issuance shall contain native methods.

OE.VERIFICATION All the bytecodes shall be verified at least once, before the loading, before the installation or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. See #.VERIFICATION for details.
Additionally, the applet shall follow all the recommendations, if any, mandated in the platform guidance for maintaining the isolation property of the platform.
Application Note: Constraints to maintain the isolation property of the platform are provided by the platform developer in application development guidance. The constraints apply to all application code loaded in the platform.

OE.CODE-EVIDENCE For application code loaded pre-issuance, evaluated technical measures implemented by the TOE or audited organizational measures must ensure that loaded application has not been changed since the code verifications required in OE.VERIFICATION.
For application code loaded post-issuance and verified off-card according to the requirements of OE.VERIFICATION, the verification authority shall provide digital evidence to the TOE that the application code has not been modified after the code verification and that he is the actor who performed code verification.
For application code loaded post-issuance and partially or entirely verified on-card, technical measures must ensure that the verification required in OE.VERIFICATION are performed. On-card bytecode verifier is out of the scope of this Protection Profile.
Application Note: For application code loaded post-issuance and verified off-card, the integrity and authenticity evidence can be achieved by electronic signature of the application code, after code verification, by the actor who performed verification.

5.3 Security Objectives Rationale

The following parts in comparison with the [PP] have been changed:

The objective of the environment in the PP OE.CARD-MANAGEMENT is changed to the objective O.CARD-MANAGEMENT of the TOE. The objective of the environment in the PP OE.SCP.IC is changed to O.IC_SUPPORT. OE.SCP.RECOVERY is replaced by O.RECOVERY and OE.SCP.SUPPORT is replaced by O.OS_SUPPORT. The (in the PP marked as) additional security objective O.EXT-MEM is added. O.RND is also included, as the hardware platform provides the TRNG.

5.3.1 Threats

5.3.1.1 Confidentiality

T.CONFID-APPLI-DATA This threat is countered by the security objective for the operational environment regarding bytecode verification (OE.VERIFICATION). It is also covered by the isolation commitments stated in the (O.FIREWALL) objective. It relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.

The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objectives O.RECOVERY and O.OS_SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

As applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys, PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) shall contribute in covering this threat by controlling the sharing of the global PIN between the applets.

Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The disclosure of such data is prevented by the security objective O.GLOBAL_ARRAYS_CONFID.

Any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the O.REALLOCATION objective. That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused.

O.EXT-MEM covers this threat as it provides controlled access means to the external memory and ensures that the external memory does not address Java Card System memory.

T.CONFID-JCS-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of those instructions enables reading a piece of code, no Java Card applet can therefore be executed to disclose a piece of code. Native applications are also harmless because of the objective O.NATIVE, so no application can be run to disclose a piece of code.

The (#.VERIFICATION) security aspect is addressed in this ST by the objective for the environment OE.VERIFICATION.

The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

O.EXT-MEM covers this threat as it provides controlled access means to the external memory and ensures that the external memory does not address Java Card System memory.

T.CONFID-JCS-DATA This threat is covered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) security objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.

The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objectives O.RECOVERY and O.OS_SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

O.EXT-MEM covers this threat as it provides controlled access means to the external memory and ensures that the external memory does not address Java Card System memory.

5.3.1.2 Integrity

T.INTEG-APPLI-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective O.NATIVE, so no application can run to modify a piece of code.

The (#.VERIFICATION) security aspect is addressed in this configuration by the objective for the environment OE.VERIFICATION.

The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that integrity and authenticity evidences exist for the application code loaded into the platform.

O.EXT-MEM covers this threat as it provides controlled access means to the external memory and ensures that the external memory does not address Java Card System memory.

T.INTEG-APPLI-CODE.LOAD This threat is countered by the security objective O.LOAD which ensures that the loading of packages is done securely and thus preserves the integrity of packages code.

The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. By controlling the access to card management functions such as the installation, update or deletion of applets the objective O.CARD-MANAGEMENT contributes to cover this threat.

T.INTEG-APPLI-DATA This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.

The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. The objectives O.RECOVERY and O.OS_SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

Concerning the confidentiality and integrity of application sensitive data, as applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys and PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) is also concerned.

Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The integrity of the information stored in that buffer is ensured by the objective O.GLOBAL_ARRAYS_INTEG.

Finally, any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the O.REALLOCATION objective. That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused.

T.INTEG-APPLI-DATA.LOAD This threat is countered by the security objective O.LOAD which ensures that the loading of packages is done securely and thus preserves the integrity of applications data.

The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. By controlling the access to card management functions such as the installation, update or deletion of applets the objective O.CARD-MANAGEMENT contributes to cover this threat.

T.INTEG-JCS-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective O.NATIVE, so no application can be run to modify a piece of code.

The (#.VERIFICATION) security aspect is addressed in this configuration by the objective for the environment OE.VERIFICATION.

The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity.

O.EXT-MEM covers this threat as it provides controlled access means to the external memory and ensures that the external memory does not address Java Card System memory.

T.INTEG-JCS-DATA This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.

The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. The objectives O.RECOVERY and O.OS_SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

O.EXT-MEM covers this threat as it provides controlled access means to the external memory and ensures that the external memory does not address Java Card System memory.

5.3.1.3 Identity Usurpation

T.SID.1 As impersonation is usually the result of successfully disclosing and modifying some assets, this threat is mainly countered by the objectives concerning the isolation of application data (like PINs), ensured by the (O.FIREWALL). Uniqueness of subject-identity (O.SID) also participates to face this threat. It should be noticed that the AIDs, which are used for applet identification, are TSF data.

In this configuration, usurpation of identity resulting from a malicious installation of an applet on the card is covered by the objective O.INSTALL.

The installation parameters of an applet (like its name) are loaded into a global array that is also shared by all the applications. The disclosure of those parameters (which could be used to impersonate the applet) is countered by the objectives O.GLOBAL_ARRAYS_CONFID and O.GLOBAL_ARRAYS_INTEG.

The objective O.CARD-MANAGEMENT contributes, by preventing usurpation of identity resulting from a malicious installation of an applet on the card, to counter this threat

T.SID.2 This is covered by integrity of TSF data, subject-identification (O.SID), the firewall (O.FIREWALL) and its good working order (O.OPERATE).

The objective O.INSTALL contributes to counter this threat by ensuring that installing an applet has no effect on the state of other applets and thus can't change the TOE's attribution of privileged roles.

The objectives O.RECOVERY and O.OS_SUPPORT are intended to support the O.OPERATE objective of the TOE, so they are indirectly related to the threats that this latter objective contributes to counter.

5.3.1.4 Unauthorized Execution

T.EXE-CODE.1 Unauthorized execution of a method is prevented by the objective OE.VERIFICATION. This threat particularly concerns the point (8) of the security aspect #VERIFICATION (access modifiers and scope of accessibility for classes, fields and methods). The O.FIREWALL objective is also concerned, because it prevents the execution of non-shareable methods of a class instance by any subject apart from the class instance owner.

T.EXE-CODE.2 Unauthorized execution of a method fragment or arbitrary data is prevented by the objective OE.VERIFICATION. This threat particularly concerns those points of the security aspect related to control flow confinement and the validity of the method references used in the bytecodes.

T.NATIVE This threat is countered by O.NATIVE which ensures that a Java Card applet can only access native methods indirectly that is, through an API. OE.APPLET also covers this threat by ensuring that no native applets shall be loaded in post-issuance. In addition to this, the bytecode verifier also prevents the program counter of an applet to jump into a piece of native code by confining the control flow to the currently executed method (OE.VERIFICATION).

5.3.1.5 Denial Of Service

T.RESOURCES This threat is directly countered by objectives on resource-management (O.RESOURCES) for runtime purposes and good working order (O.OPERATE) in a general manner.

Consumption of resources during installation and other card management operations are covered, in case of failure, by O.INSTALL.

It should be noticed that, for what relates to CPU usage, the Java Card platform is single-threaded and it is possible for an ill-formed application (either native or not) to monopolize the CPU. However, a smart card can be physically interrupted (card removal or hardware reset) and most CADs implement a timeout policy that prevent them from being blocked should a card fails to answer. That point is out of scope of this Protection Profile, though.

Finally, the objectives O.RECOVERY and O.OS_SUPPORT are intended to support the O.OPERATE and O.RESOURCES objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

5.3.1.6 Card Management

T.DELETION This threat is covered by the O.DELETION security objective which ensures that both applet and package deletion perform as expected.

The objective O.CARD-MANAGEMENT controls the access to card management functions and thus contributes to cover this threat.

T.INSTALL This threat is covered by the security objective O.INSTALL which ensures that the installation of an applet performs as expected and the security objectives O.LOAD which ensures that the loading of a package into the card is safe.

The objective O.CARD-MANAGEMENT controls the access to card management functions and thus contributes to cover this threat.

T.COMMUNICATION This threat is covered by the security objective O.COMMUNICATION security objective which authenticates the origin of the card management.

The objective O.CARD-MANAGEMENT controls the access to card management functions and thus contributes to cover this threat.

5.3.1.7 Services

T.OBJ-DELETION This threat is covered by the O.OBJ-DELETION security objective which ensures that object deletion shall not break references to objects.

5.3.1.8 Miscellaneous

T.PHYSICAL Covered by O.IC_SUPPORT. Physical protections rely on the underlying platform.
The objective O.RND controls the cryptographic quality of random number generation which covers together with O.IC_SUPPORT the point (7) of the security aspect #.SCP, extracting cryptographic keys.

5.3.1.9 Threats Composite Platform

T.LEAKAGE O.RND ensures the cryptographic quality of random number generation, which counters the threat. O.IC_SUPPORT covers the threat as the IC is tamper resistant.

T.FAULT O.IC_SUPPORT covers the threat with the IC security features against physical attacks. O.RND ensures the cryptographic quality of random number generation, which counters the threat.

T.RND Here the attacker is expected to take advantage of statistical properties of the random numbers generated by the TOE without specific knowledge about the TOE's generator. Malfunctions or premature ageing are also considered which may assist in getting information about random numbers. This is directly covered by O.RND and O.IC_SUPPORT covers the threat with the IC security features against physical attacks.

5.3.2 Organizational Security Policies

OSP.VERIFICATION This policy is upheld by the security objective of the environment OE.VERIFICATION which guarantees that all the bytecodes shall be verified at least once, before the loading, before the installation or before the execution in order to ensure that each bytecode is valid at execution time.

This policy is also upheld by the security objective of the environment OE.CODE-EVIDENCE which ensures that evidences exist that the application code has been verified and not changed after verification.

5.3.3 Assumptions

A.APPLET This assumption is upheld by the security objective for the operational environment OE.APPLET which ensures that no applet loaded post-issuance shall contain native methods.

A.VERIFICATION This assumption is upheld by the security objective on the operational environment OE.VERIFICATION which guarantees that all the bytecodes shall be verified at least once, before the loading, before the installation or before the execution in order to ensure that each bytecode is valid at execution time.
This assumption is also upheld by the security objective of the environment OE.CODE-EVIDENCE which ensures that evidences exist that the application code has been verified and not changed after verification.

5.3.4 SPD and Security Objectives

Threats	Security Objectives	Rationale
T.CONFID-APPLI-DATA	O.RECOVERY , O.OS_SUPPORT, O.CARD-MANAGEMENT , OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.REALLOCATION, O.EXT-MEM	**Section 5.3.1**
T.CONFID-JCS-CODE	OE.VERIFICATION, O.CARD-MANAGEMENT , O.NATIVE, O.EXT-MEM	**Section 5.3.1**
T.CONFID-JCS-DATA	O.RECOVERY , O.OS_SUPPORT, O.CARD-MANAGEMENT , OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM, O.EXT-MEM	**Section 5.3.1**
T.INTEG-APPLI-CODE	O.CARD-MANAGEMENT , OE.VERIFICATION, O.NATIVE, OE.CODE-EVIDENCE, O.EXT-MEM	**Section 5.3.1**
T.INTEG-APPLI-CODE.LOAD	O.LOAD, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	**Section 5.3.1**
T.INTEG-APPLI-DATA	O.RECOVERY , O.OS_SUPPORT, O.CARD-MANAGEMENT , OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.GLOBAL_ARRAYS_INTEG, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.REALLOCATION, OE.CODE-EVIDENCE	**Section 5.3.1**
T.INTEG-APPLI-DATA.LOAD	O.LOAD, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	**Section 5.3.1**
T.INTEG-JCS-CODE	O.CARD-MANAGEMENT , OE.VERIFICATION, O.NATIVE, OE.CODE-EVIDENCE, O.EXT-MEM	**Section 5.3.1**
T.INTEG-JCS-DATA	O.RECOVERY , O.OS_SUPPORT, O.CARD-MANAGEMENT , OE.VERIFICATION, O.SID,	**Section 5.3.1**

	O.OPERATE, O.FIREWALL, O.ALARM, OE.CODE-EVIDENCE, O.EXT-MEM	
T.SID.1	O.CARD-MANAGEMENT , O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.INSTALL, O.SID	**Section 5.3.1**
T.SID.2	O.RECOVERY , O.OS_SUPPORT, O.SID, O.OPERATE, O.FIREWALL, O.INSTALL	**Section 5.3.1**
T.EXE-CODE.1	OE.VERIFICATION, O.FIREWALL	**Section 5.3.1**
T.EXE-CODE.2	OE.VERIFICATION	**Section 5.3.1**
T.NATIVE	OE.VERIFICATION, OE.APPLET, O.NATIVE	**Section 5.3.1**
T.RESOURCES	O.INSTALL, O.OPERATE, O.RESOURCES, O.RECOVERY , O.OS_SUPPORT	**Section 5.3.1**
T.DELETION	O.DELETION, O.CARD-MANAGEMENT	**Section 5.3.1**
T.INSTALL	O.INSTALL, O.LOAD, O.CARD-MANAGEMENT	**Section 5.3.1**
T.COMMUNICATION	O.CARD-MANAGEMENT, O.COMMUNICATION	**Section 5.3.1**
T.OBJ-DELETION	O.OBJ-DELETION	**Section 5.3.1**
T.PHYSICAL, T.LEAKAGE, T.FAULT, T.RND	O.IC_SUPPORT, O.RND	**Section 7.4**

Table 2 Threats and Security Objectives - Coverage

Security Objectives	Threats
O.SID	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.1, T.SID.2
O.FIREWALL	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.1, T.SID.2, T.EXE-CODE.1
O.GLOBAL_ARRAYS_CONFID	T.CONFID-APPLI-DATA, T.SID.1
O.GLOBAL_ARRAYS_INTEG	T.INTEG-APPLI-DATA, T.SID.1
O.NATIVE	T.CONFID-JCS-CODE, T.INTEG-APPLI-CODE, T.INTEG-JCS-CODE, T.NATIVE
O.OPERATE	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES
O.REALLOCATION	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.RESOURCES	T.RESOURCES
O.ALARM	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA
O.CIPHER	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.KEY-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.PIN-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.TRANSACTION	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.OBJ-DELETION	T.OBJ-DELETION
O.DELETION	T.DELETION
O.LOAD	T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA.LOAD, T.INSTALL

Security Objectives	Threats
O.INSTALL	T.SID.1, T.SID.2, T.RESOURCES, T.INSTALL
O.COMMUNICATION	T.COMMUNICATION
OE.APPLET	T.NATIVE
O.CARD-MANAGEMENT	T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA, T.INTEG-APPLI-DATA.LOAD, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.SID.1, T.DELETION, T.INSTALL, T.COMMUNICATION
O.IC_SUPPORT	T.PHYSICAL
O.RECOVERY	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES
O.OS_SUPPORT	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES
O.EXT-MEM	T.CONFID-JCS-CODE, T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA
O.RND	T.RND, T.PHYSICAL, T.LEAKAGE, T.FAULT
OE.VERIFICATION	T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-DATA, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.EXE-CODE.1, T.EXE-CODE.2, T.NATIVE
OE.CODE-EVIDENCE	T.INTEG-APPLI-CODE, T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA, T.INTEG-APPLI-DATA.LOAD, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA

Table 3 Security Objectives and Threats - Coverage

Organizational Security Policies	Security Objectives
OSP.VERIFICATION	O.LOAD, OE.VERIFICATION, OE.CODE-EVIDENCE

Table 4 OSPs and Security Objectives - Coverage

Security Objectives	Organizational Security Policies
O.LOAD	OSP.VERIFICATION
OE.VERIFICATION	OSP.VERIFICATION
OE.CODE-EVIDENCE	OSP.VERIFICATION

Table 5 Security Objectives and OSPs - Coverage

Assumptions	Security objectives for the Operational Environment
A.APPLET	OE.APPLET
A.VERIFICATION	OE.VERIFICATION, OE.CODE-EVIDENCE

Table 6 Assumptions and Security Objectives for the Operational Environment - Coverage

Security Objectives for the Operational Environment	Assumptions
OE.APPLET	A.APPLET
OE.VERIFICATION	A.VERIFICATION
OE.CODE-EVIDENCE	A.VERIFICATION

Table 7 Security Objectives for the Operational Environment and Assumptions - Coverage

6 Security Requirements

The operations of the SFR, that are left open in the PP (assignment, iteration, selection, and refinement), are printed in **bold type** and marked with brackets together with the operation, e.g. [**assignment**:…]. The operations that are completed in the PP are copied with no further marks.

6.1 TOE Security Functional Requirements

Group	Description
Core with Logical Channels (*CoreG_LC*)	The CoreG_LC contains the requirements concerning the runtime environment of the Java Card System implementing logical channels. This includes the firewall policy and the requirements related to the Java Card API. Logical channels are a Java Card specification version 2.2 feature. This group is the union of requirements from the Core (*CoreG*) and the Logical channels (*LCG*) groups defined in [PP/0305] (cf. Java Card System Protection Profile Collection [PP JCS]).
Installation (InstG)	The InstG contains the security requirements concerning the installation of post-issuance applications. It does not address card management issues in the broad sense, but only those security aspects of the installation procedure that are related to applet execution.
Applet deletion (ADELG)	The ADELG contains the security requirements for erasing installed applets from the card, a feature introduced in Java Card specification version 2.2.
Object deletion (ODELG)	The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism. This is a Java Card specification version 2.2 feature.
Secure carrier (CarG)	The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing, in those configurations that do not support on-card static or dynamic bytecode verification, the installation of a package that has not been

	bytecode verified, or that has been modified after bytecode verification.
External memory (EMG)	The EMG group contains the requirements of the Extended Memory feature. It is an API-based mechanism to access the external memory outside the addressable Java Card VM space. It covers the OS Extension architecture feature (Sandboxing) which allows extension of the OS without the need to recompile and deploy a new JCS image.
GlobalPlatform (GPG)	The GPG contains the requirements that allow defining a policy for controlling access to card content management operations.
Smart card platform (SCPG)	The SCPG contains the security requirements for the smart card platform.

Table 8 Security Functional Requirements overview

This section states the security functional requirements for the Java Card System - Open configuration. For readability and for compatibility with the original Java Card System Protection Profile Collection - Standard 2.2 Configuration [PP/0305], requirements are arranged into groups. All the groups defined in the table below apply to this Protection Profile.

The SFRs refer to all potentially applicable subjects, objects, information, operations, and security attributes. The TOE does not provide JCRMI functionality.

Subjects are active components of the TOE that (essentially) act on the behalf of users. The users of the TOE include people or institutions (for this TOE the **Chip Manufacturer, Composite Product Integrator**, **Applet Developer, Card Issuer, Verification Authority**), hardware (like the CAD where the card is inserted or the PCD) and software components (like the application packages installed on the card). Some of the users may just be aliases for other users. For instance, the verification authority in charge of the bytecode verification of the applications is just an alias for the card issuer for this TOE.

Subjects (prefixed with an "S") are described in the following table:

Subject	Description
S.ADEL	The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([JCRE3], §11), but its role asks anyway for a specific treatment from the security viewpoint. This subject is unique and is involved in the ADEL security policy defined in §6.1.3
S.APPLET	Any applet instance.
S.BCV	The bytecode verifier (BCV), which acts on behalf of the verification authority who is in charge of the bytecode verification of the packages. This subject is involved in the PACKAGE LOADING security policy defined in §6.1.7.
S.CAD	The CAD represents off-card entity that communicates with the S.INSTALLER.
S.INSTALLER	The installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the loading of packages and installation of applets.
S.GPINST	The GP installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the PACKAGE LOADING security policy defined in §6.1.7.
S.JCRE	The runtime environment under which Java programs in a smart card are executed.
S.JCVM	The bytecode interpreter that enforces the firewall at runtime.
S.LOCAL	Operands stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.
S.MEMBER	Any object's field, static field or array position.
S.PACKAGE	A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of Java Card technology, it defines either a user library, or one or several applets.

Table 9 Subjects

Objects (prefixed with an "O") are described in the following table:

Object	Description
O.APPLET	Any installed applet, its code and data.
O.CODE_PKG	The code of a package, including all linking information. On the Java Card platform, a package is the installation unit.
O.JAVAOBJECT	Java class instance or array. It should be noticed that KEYS, PIN, arrays and applet instances are specific objects in the Java programming language.
O.EXT_MEM_INSTANCE	Any External Memory Instance created from the MemoryAccess Interface of the Java Card API [JCAPI3].

Table 10 Objects

Information (prefixed with an "I") is described in the following table:

Information	Description
I.APDU	Any APDU sent to or from the card through the communication channel.
I.DATA	JCVM Reference Data: objectref addresses of APDU buffer, JCRE-owned instances of APDU class and byte array for install method.

Table 11 Information

Security attributes linked to these subjects, objects and information are described in the following table with their values:

Security attribute	Description/Value
Active Applets	The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels.
Address space	Accessible memory portion.
Applet Selection Status	"Selected" or "Deselected".

Applet's version number	The version number of an applet (package) indicated in the export file.
Class	Identifies the implementation class of the remote object.
Context	Package AID or "Java Card RE".
Currently Active Context	Package AID or "Java Card RE".
Dependent package AID	Allows the retrieval of the Package AID and Applet's version number ([JCVM3], §4.5.2).
LC Selection Status	Multiselectable, Non-multiselectable or "None".
LifeTime	CLEAR_ON_DESELECT or PERSISTENT (*).
Owner	The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the package). The owner of a remote object is the applet instance that created the object.
Package AID	The AID of each package indicated in the export file.
Registered Applets	The set of AID of the applet instances registered on the card.
Resident Packages	The set of AIDs of the packages already loaded on the card.
Selected Applet Context	Package AID or "None".
Sharing	Standards, SIO, Java Card RE entry point or global array.
Static References	Static fields of a package may contain references to objects. The Static References attribute records those references.

Table 12 Security attributes

(*) Transient objects of type CLEAR_ON_RESET behave like persistent objects in that they can be accessed only when the Currently Active Context is the object's context.

Operations (prefixed with "OP") are described in the following table. Each operation has parameters given between brackets, among which there is the "accessed object", the first one, when applicable. Parameters may be seen as security attributes that are under the control of the subject performing the operation.

Operation	Description
OP.ARRAY_ACCESS(O.JAVAOBJECT, field)	Read/Write an array component.
OP.CREATE(Sharing, LifeTime) (*)	Creation of an object (new or makeTransient call).
OP.LOAD_PCKG(O.PACKAGE, package AID, load parameters, ...)	Load and link a package from the S.CAD into card NVRAM.
OP. INSTALL_APPLET (O.PACKAGE, O.APPLET, application AID, application privileges, ...)	Install and create a selectable applet instance from an installed package with a specific application AID, application privileges and install parameters.
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed applet and its objects, either logically or physically.
OP.DELETE_PCKG(O.CODE_PKG,...)	Delete a package, either logically or physically.
OP.DELETE_PCKG_APPLET(O.CODE_PKG,...)	Delete a package and its installed applets, either logically or physically.

OP.GP(…)	GlobalPlatform's card content management APDU commands and API methods (defined in Appendix A of [GP]):
	loading (Section 9.3.5 of [GP]);
	installation (Section 9.3.6 of [GP]);
	extradition (Section 9.4.1 of [GP]);
	registry update (Section 9.4.2 of [GP]);
	content removal (Section 9.5 of [GP])]

OP.INSTANCE_FIELD(O.JAVAOBJECT, field)	Read/Write a field of an instance of a class in the Java programming language.
OP.INVK_VIRTUAL(O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object).
OP.INVK_INTERFACE(O.JAVAOBJECT, method, arg1,...)	Invoke an interface method.
OP.JAVA(...)	Any access in the sense of [JCRE3], §6.2.8. It stands for one of the operations OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW, OP.TYPE_ACCESS.
OP.PUT(S1,S2,I)	Transfer a piece of information I from S1 to S2.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see [JCRE3], §6.2.8.7).
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	Invoke checkcast or instanceof on an object in order to access to classes (standard or shareable interfaces objects).
OP.CREATE_EXT_MEM_INSTANCE	Creation of an instance of the MemoryAccess Interface.
OP.READ_EXT_MEM(O.EXT_MEM_INSTANCE, address)	Reading the external memory.
OP.WRITE_EXT_MEM(O.EXT_MEM_INSTANCE, address)	Writing the external memory.

Table 13 Operations

(*) For this operation, there is no accessed object. This rule enforces that shareable transient objects are not allowed. For instance, during the creation of an object, the JavaCardClass attribute's value is chosen by the creator.

6.1.1 Coreg_LC Security Functional Requirements

This group is focused on the main security policy of the Java Card System, known as the firewall.

6.1.1.1 FIREWALL POLICY

FDP_ACC.2/FIREWALL Complete access control

FDP_ACC.2.1/FIREWALL The TSF shall enforce the **FIREWALL access control SFP** on **S.PACKAGE, S.JCRE, S.JCVM, O.JAVAOBJECT** and all operations among subjects and objects covered by the SFP.

Refinement:

The operations involved in the policy are:

- OP.CREATE,
- OP.INVK_INTERFACE,
- OP.INVK_VIRTUAL,
- OP.JAVA,
- OP.THROW,
- OP.TYPE_ACCESS.

FDP_ACC.2.2/FIREWALL The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

Application note:

It should be noticed that accessing array's components of a static array, and more generally fields and methods of static objects, is an access to the corresponding O.JAVAOBJECT.

FDP_ACF.1/FIREWALL Security attribute based access control

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

Subject/Object	Security attributes
S.PACKAGE	LC Selection Status
S.JCVM	Active Applets, Currently Active Context
S.JCRE	Selected Applet Context
O.JAVAOBJECT	Sharing, Context, LifeTime

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:

- **R.JAVA.1 ([JCRE3], §6.2.8): S.PACKAGE may freely perform OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW or OP.TYPE_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".**
- **R.JAVA.2 ([JCRE3], §6.2.8): S.PACKAGE may freely perform OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE or OP.THROW upon any O.JAVAOBJECT whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if O.JAVAOBJECT's Context attribute has the same value as the active context.**
- **R.JAVA.3 ([JCRE3], §6.2.8.10): S.PACKAGE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT whose Sharing attribute has value "SIO" only if O.JAVAOBJECT is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.**
- **R.JAVA.4 ([JCRE3], §6.2.8.6): S.PACKAGE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if the invoked interface method extends the Shareable interface and one of the following conditions applies:**

a) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable",

b) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Non-multiselectable", and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute Active Applets.

- **R.JAVA.5: S.PACKAGE may perform OP.CREATE only if the value of the Sharing parameter is "Standard"**.

FDP_ACF.1.3/FIREWALL The TSF shall explicitly authorize access of subjects to objects based on the following additional rules:

- **1) The subject S.JCRE can freely perform OP.JAVA(") and OP.CREATE, with the exception given in FDP_ACF.1.4/FIREWALL, provided it is the Currently Active Context.**
- **2) The only means that the subject S.JCVM shall provide for an application to execute native code is the invocation of a Java Card API method (through OP.INVK_INTERFACE or OP.INVK_VIRTUAL)**.

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

- **1) Any subject with OP.JAVA upon an O.JAVAOBJECT whose LifeTime attribute has value "CLEAR_ON_DESELECT" if O.JAVAOBJECT's Context attribute is not the same as the Selected Applet Context.**
- **2) Any subject attempting to create an object by the means of OP.CREATE and a "CLEAR_ON_DESELECT" LifeTime parameter if the active context is not the same as the Selected Applet Context**

Application note:

FDP_ACF.1.4/FIREWALL:

- The deletion of applets render some O.JAVAOBJECT inaccessible, and the Java Card RE is in charge of this aspect.

  The implementation is designed in such a way that deletion of installed executable content like applets or packages does not introduce security holes in the form of broken references to garbage collected code or data, nor does it affect the integrity or confidentiality of remaining applets in any way. All static references are detected prior to a deletion and if present, delete operation is canceled with an applet left in place, rather than leaving a static reference dangling such that a newly loaded applet’s memory segment could be compromised by that dangling reference.

Upon erase event, any data owned by the deleted applet is no longer accessible and as such deleted applet cannot be selected or receive APDU commands. In fact, Card Manager can no long locate applet’s select() or process() Java methods because all such relevant information has been physically removed from applet’s registry. Naturally, package deletion physically removes its entry from on-card registry and makes its opcodes no longer available for execution by Java Card VM.

Transactions are used throughout delete sequence to assure that any power failures do not corrupt the deletion process and jeopardize the TOE’s internal data structures

In the case of an array type, fields are components of the array ([JVM], §2.14, §2.7.7), as well as the length; the only methods of an array object are those inherited from the Object class.

The Sharing attribute defines four categories of objects:

- Standard ones, whose both fields and methods are under the firewall policy,
- Shareable interface Objects (SIO), which provide a secure mechanism for inter-applet communication,
- JCRE entry points (Temporary or Permanent), who have freely accessible methods but protected fields,
- Global arrays, having both unprotected fields (including components; refer to JavaCardClass discussion above) and methods.

When a new object is created, it is associated with the Currently Active Context. But the object is owned by the applet instance within the Currently Active Context when the object is instantiated ([JCRE3], §6.1.3). An object is owned by an applet instance, by the JCRE or by the package library where it has been defined (these latter objects can only be arrays that initialize static fields of packages).

([JCRE3], Glossary) Selected Applet Context. The Java Card RE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command with this applet's AID, the Java Card RE makes this applet the Selected Applet Context. The Java Card RE sends all APDU commands to the Selected Applet Context.

While the expression "Selected Applet Context" refers to a specific installed applet, the relevant aspect to the policy is the context (package AID) of the selected applet. In this policy, the "Selected Applet Context" is the AID of the selected package.

([JCRE3], §6.1.2.1) At any point in time, there is only one active context within the Java Card VM (this is called the Currently Active Context).

It should be noticed that the invocation of static methods (or access to a static field) is not considered by this policy, as there are no firewall rules. They have no effect on the active context as well and the "acting package" is not the one to which the static method belongs to in this case.

It should be noticed that the Java Card platform, version 2.2.x and version 3 Classic Edition, introduces the possibility for an applet instance to be selected on multiple logical channels at the same time, or accepting other applets belonging to the same package being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same package are either all multiselectable or not ([JCVM3], §2.2.5). Therefore, the selection mode can be regarded as an attribute of packages. No selection mode is defined for a library package.

An applet instance will be considered an active applet instance if it is currently selected in at least one logical channel. An applet instance is the currently selected applet instance only if it is processing the current command. There can only be one currently selected applet instance at a given time. ([JCRE3], §4).

FDP_IFC.1/JCVM Subset information flow control

FDP_IFC.1.1/JCVM The TSF shall enforce the **JCVM information flow control SFP** on **S.JCVM, S.LOCAL, S.MEMBER, I.DATA and OP.PUT(S1, S2, I)**.

Application note:

It should be noticed that references of temporary Java Card RE entry points, which cannot be stored in class variables, instance variables or array components, are transferred from the internal memory of the Java Card RE (TSF data) to some stack through specific APIs (Java Card RE owned exceptions) or Java Card RE invoked methods (such as the process(APDU apdu)); these are causes of OP.PUT(S1,S2,I) operations as well.

FDP_IFF.1/JCVM Simple security attributes

FDP_IFF.1.1/JCVM The TSF shall enforce the **JCVM information flow control SFP** based on the following types of subject and information security attributes:

Subjects	Security attributes
S.JCVM	Currently Active Context

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

- **An operation OP.PUT(S1, S.MEMBER, I.DATA) is allowed if and only if the Currently Active Context is "Java Card RE";**
- **other OP.PUT operations are allowed regardless of the Currently Active Context's value**.

FDP_IFF.1.3/JCVM The TSF shall enforce the **[assignment: none]**

FDP_IFF.1.4/JCVM The TSF shall explicitly authorize an information flow based on the following rules: **[assignment: none]**

FDP_IFF.1.5/JCVM The TSF shall explicitly deny an information flow based on the following rules: **[assignment: none]**

Application Note:

The storage of temporary Java Card RE-owned objects references is runtime-enforced ([JCRE3], §6.2.8.1-3).

It should be noticed that this policy essentially applies to the execution of bytecode. Native methods, the Java Card RE itself and possibly some API methods can be granted specific rights or limitations through the FDP_IFF.1.3/JCVM to FDP_IFF.1.5/JCVM elements. The way the Java Card virtual machine manages the transfer of values on the stack and local variables (returned values, uncaught exceptions) from and to internal registers is implementation-dependent. For instance, a returned reference, depending on the implementation of the stack frame, may transit through an internal register prior to being pushed on the stack of the invoker. The returned bytecode would cause more than one OP.PUT operation under this scheme.

FDP_RIP.1/OBJECTS Subset residual information protection

FDP_RIP.1.1/OBJECTS The TSF shall ensure that any previous information content of a resource is made unavailable upon the **allocation of the resource to** the following objects: **class instances and arrays**.

Application Note:

The semantics of the Java programming language requires for any object field and array position to be initialized with default values when the resource is allocated [JVM], §2.5.1.

FMT_MSA.1/JCRE Management of security attributes

FMT_MSA.1.1/JCRE The TSF shall enforce the **FIREWALL access control SFP** to restrict the ability to **modify** the security attributes **Selected Applet Context** to **the Java Card RE**.

Application note:

The modification of the Selected Applet Context are performed in accordance with the rules given in [JCRE3], §4 and [JCVM3], §3.4.

FMT_MSA.1/JCVM Management of security attributes

FMT_MSA.1.1/JCVM The TSF shall enforce the **FIREWALL access control SFP and the JCVM information flow control SFP** to restrict the ability to **modify** the security attributes **Currently Active Context and Active Applets** to **the Java Card VM (S.JCVM)**.

Application Note:

The modification of the Currently Active Context are performed in accordance with the rules given in [JCRE3], §4 and [JCVM3], §3.4.

FMT_MSA.2/FIREWALL_JCVM Secure security attributes

FMT_MSA.2.1/FIREWALL_JCVM The TSF shall ensure that only secure values are accepted for **all the security attributes of subjects and objects defined in the FIREWALL access control SFP and the JCVM information flow control SFP**.

Application note:

The following rules are implemented in the TOE. The TOE does not support the creation of transient objects belonging to arbitrary classes.

- The Context attribute of an O.JAVAOBJECT must correspond to that of an installed applet or be "Java Card RE".
- An O.JAVAOBJECT whose Sharing attribute is a Java Card RE entry point or a global array necessarily has "Java Card RE" as the value for its Context security attribute.
- An O.JAVAOBJECT whose Sharing attribute value is a global array necessarily has "array of primitive type" as a JavaCardClass security attribute's value.
- Any O.JAVAOBJECT whose Sharing attribute value is not "Standard" has a PERSISTENT-LifeTime attribute's value.
- Any O.JAVAOBJECT whose LifeTime attribute value is not PERSISTENT has an array type as JavaCardClass attribute's value.

FMT_MSA.3/FIREWALL Static attribute initialization

FMT_MSA.3.1/FIREWALL The TSF shall enforce the **FIREWALL access control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/FIREWALL [Editorially Refined] The TSF shall not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

Application note:

FMT_MSA.3.1/FIREWALL

- Objects' security attributes of the access control policy are created and initialized at the creation of the object or the subject. Afterwards, these attributes are no longer mutable (FMT_MSA.1/JCRE). At the creation of an object (OP.CREATE), the newly created object, assuming that the FIREWALL access control SFP permits the operation, gets its Lifetime and

Sharing attributes from the parameters of the operation; on the contrary, its Context attribute has a default value, which is its creator's Context attribute and AID respectively ([JCRE3], §6.1.3). There is one default value for the Selected Applet Context that is the default applet identifier's Context, and one default value for the Currently Active Context that is "Java Card RE".

- The knowledge of which reference corresponds to a temporary entry point object or a global array and which does not is solely available to the Java Card RE (and the Java Card virtual machine).

FMT_MSA.3.2/FIREWALL

- The intent is that none of the identified roles has privileges with regard to the default values of the security attributes. It should be noticed that creation of objects is an operation controlled by the FIREWALL access control SFP. The operation shall fail anyway if the created object would have had security attributes whose value violates FMT_MSA.2.1/FIREWALL_JCVM.

FMT_MSA.3/JCVM Static attribute initialization

FMT_MSA.3.1/JCVM The TSF shall enforce the **JCVM information flow control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/JCVM [Editorially Refined] The TSF shall not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

FMT_SMF.1 Specification of Management Functions

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

- **modify the Currently Active Context**, **the Selected Applet Context and the Active Applets**

FMT_SMR.1 Security roles

FMT_SMR.1.1 The TSF shall maintain the roles**:**

- **Java Card RE (JCRE),**

- **Java Card VM (JCVM)**.

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

6.1.1.2 Application Programming Interface

The following SFRs are related to the Java Card API and the extensions.

The whole set of cryptographic algorithms is generally not implemented because of limited memory resources and/or limitations due to exportation. Therefore, the following requirements only apply to the implemented subset.

It should be noticed that the execution of the additional native code is not within the TSF. Nevertheless, access to API native methods from the Java Card System is controlled by TSF because there is no difference between native and interpreted methods in their interface or invocation mechanism.

FCS_CKM.1 Cryptographic key generation

FCS_CKM.1.1/RSA The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **[assignment: key generation algorithm for RSA (with or without CRT)]** and specified cryptographic key sizes **[assignment: RSA (with or without CRT): 1984 - 2048 bits, in increments of 32 bits]** that meet the following: **[assignment: According to section 3.2(2) in [PKCS V2.1], for u=2, i.e., without any (r_i, d_i, t_i), i > 2. For p x q < $2^{2048+64}$ additionally according to section 3.2(1)]**.

FCS_CKM.1.1/EC The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **[assignment: key generation for EC]** and specified cryptographic key sizes **[assignment: 192, 224, 233, 256, 283, 320, 384, 409, 512, 521]** that meet the following: **[assignment:**

ECDSA Key Generation:

1. **According to the appendix A4.3 in [ANSI X9.62-2005] (the cofactor h is not supported),**

2. According to section 6.4.2 Generation of signature key and verification key in [ISO/IEC 14888-3:2009]

3. According to appendix A.16.9 An algorithm for generating EC keys in [IEEE Std. 1363:2000]

Note the former ISO/IEC standard 15946 has been withdrawn

Application note:

- The keys can be generated and diversified in accordance with [JCAPI3] specification in classes KeyBuilder and KeyPair (at least Session key generation).
- This component is instantiated according to the version of the Java Card API applying to this security target and the implemented algorithms ([JCAPI3]).
- The certification covers the NIST standard FIPS 186-4 Elliptic Curves P-{192,224,256,384,521} and Brainpool RFC 5639 Elliptic Curves brainpoolIP {192,224,256,320,384,512}r1 and brainpoolIP{192,224,256,320,384,512}t1 and Curve B-{233 283, 409} and Curve K-{233,283,409}.
- The RSA, RSA KeyGen and the ECC modules are optional modules delivery options for the TOE. The TOE can come without the RSA, RSA KeyGen or ECC module. In that case it does not provide the Specific Security Functionality RSA or ECC. In the absence of the RSA KeyGen module, RSA key generation functionality is not available in the TOE.

FCS_CKM.2 Cryptographic key distribution

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **[assignment: setKey for DES and AES, setExponent**

and setModulus for RSA, as well as, setA, setB, setFieldFP, setG, setK, and setR for EC] that meets the following: **[assignment: JCAPI3]**.

Application note:

- Command SetKEY that meets [JCAPI3] specification.
- This component is instantiated according to the version of the Java Card API applying to this security target and the implemented algorithms ([JCAPI3]).

FCS_CKM.3 Cryptographic key access

FCS_CKM.3.1 The TSF shall perform **[assignment: management of DES, EC, and RSA, AES-keys]** in accordance with a specified cryptographic key access method **[assignment: methods/commands defined in packages javacard.security and javacardx.crypto]** that meets the following: **[assignment: JCAPI3]**.

Application note:

- The keys can be accessed as specified in [JCAPI3] Key class.
- This component is instantiated according to the version of the Java Card API applicable to this security target and the implemented algorithms ([JCAPI3]).
- The RSA and the ECC modules are optional modules delivery options for the TOE. The OE can come without the RSA or ECC module. In that case it does not provide the Specific Security Functionality RSA or ECC.

FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **[assignment: physically overwriting the keys with zeros]** that meets the following: **[assignment: [FIPS140]]**.

Application note:

- The keys are reset as specified in [JCAPI3] Key class, with the method clearKey(). Any access to a cleared key for ciphering or signing shall throw an exception.
- This component is instantiated according to the version of the Java Card API applicable to this security target and the implemented algorithms ([JCAPI3]).

FCS_COP.1 Cryptographic operation

FCS_COP.1.1/DES The TSF shall perform **[assignment: data encryption and decryption]** in accordance with a specified cryptographic algorithm **[assignment: TDES in ECB/CBC Mode]** and cryptographic key sizes **[assignment: 112 bits, 168 bits]** that meet the following: **[assignment: [SP800-67], [SP800-38A]]**.

FCS_COP.1.1/SHA The TSF shall perform **[assignment: hash-value value calculation of user chosen data]** in accordance with a specified cryptographic algorithm **[assignment: SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512)]** and cryptographic key sizes **[assignment: none]** that meet the following: **[assignment: [FIPS 180-4]].**

FCS_COP.1.1/RSA The TSF shall perform **[assignment: data encryption/decryption and signature generation/verification]** in accordance with a specified cryptographic algorithm **[assignment: RSA]** and cryptographic key sizes **[assignment: 1984 - 2048 bits, in increments of 32 bits]** that meet the following: **[assignment:**

Encryption: RSAEP (section 5.1.1 in PKCS v2.1, without 5.1.1.1);

Decryption (with or without CRT): RSADP (section 5.1.2 in PKCS v2.1 for u = 2, only supported up to n < $2^{2048+64}$);

Signature Generation (with or without CRT): RSASP1 (section 5.2.1 in PKCS v2.1 for u = 2 , only supported up to n < $2^{2048+64}$) and RSASSA-PKCS-v1_5 (section 8.2.1 in PKCS v2.1);

Signature Verification: RSAVP1 (section 5.2.2 in PKCS v2.1 - without 5.2.2.1) and RSASSA-PKCS-v1_5 (section 8.2.2 in PKCS v2.1).].

FCS_COP.1.1/ECDH The TSF shall perform **[assignment: elliptic curve Diffie-Hellman key agreement]** in accordance with a specified cryptographic algorithm **[assignment: ECDH]** and cryptographic key sizes **[assignment: 192, 224, 233, 256, 283, 320, 384, 409, 512, 521bits]** that meet the following**: [assignment: 1. According to section 5.4.1 in [ANSI X9.63-2001] Unlike section 5.4.1.3 our implementation not only returns the x-coordinate of the shared secret, but rather the x-coordinate and y-coordinate; 2. According to section Appendix D.6 Key agreement of Diffie-Hellman type in [ISO/IEC 11770-3:2009] the function enables the operations described in appendix D.6; 3. According to section 7.2.1 ECSVHDP-DP in [IEEE Std. 1363:2000] Unlike section**

7.2.1 our implementation not only returns the x-coordinate of the shared secret, but rather the x-coordinate and the y-coordinate.

Application note:

- The certification covers the NIST standard FIPS 186-4 Elliptic Curves P-{192,224,256,384,521} and Brainpool RFC 5639 Elliptic Curves brainpoolIP {192,224,256,320,384,512}r1 and brainpoolIP{192,224,256,320,384,512}t1 and Curve B-{233 283, 409} and Curve K-{233,283,409}.

FCS_COP.1.1/AES The TSF shall perform **[assignment: data encryption and decryption]** in accordance with a specified cryptographic algorithm **[assignment: AES in ECB/CBC mode]** and cryptographic key sizes **[assignment: 128, 192, or 256 bits]** that meet the following**: [assignment: [FIPS 197], [SP800-38A]]**.

FCS_COP.1.1/ECDSA The TSF shall perform **[assignment: digital signature generation and verification]** in accordance with a specified cryptographic algorithm **[assignment: ECDSA-FP, ECDSA-F2M]** and cryptographic key sizes **[assignment: FP: 192, 224, 256, 320, 384, 512, 521 Bit; F2M: 233, 283, 409 bits]** that meet the following: **[assignment:**

Signature Generation: 1. According to section 7.3 in [ANSI X9.62-2005] Not implemented is step d) and e) thereof. The output of step e) has to be provided as input to our function by the caller. Deviation of step c) and f): The jumps to step a) were substituted by a return of the function with an error code, the jumps are emulated by another call to our function; 2. According to sections 6.4.3 Signature Process in [ISO/IEC 14888-3:2009] Chapter 6.4.3.3 is not supported Chapter 6.4.3.5 is not supported - the hash-code of H of the message has to be provided by the caller as input for our function. Chapter 6.4.3.7 is not supported Chapter 6.4.3.8 is not supported; 3. According to section 7.2.7 ECSP-DSA in [IEEE Std. 1363:2000] Deviation of step (3) and (4): - The jumps to step 1 were substituted by a return of the function with an error code, the jumps are emulated by another call to our function

Signature Verification: 1. According to section 7.4.1 in [ANSI X9.62–2005] Not implemented is step b) and c) thereof. The output of step c) has to be provided as input to our function by the caller. Deviation of step d): Beside noted calculation, our algorithm adds a random multiple of BasepointerOrder n to the calculated values u1 and u2; 2. According to sections 6.4.4 Signature Verification Process in [ISO/IEC 14888-3:2009] Chapter 6.4.4.2 is not supported, Chapter 6.4.4.3 is not supported: -

The hash-code H of the message has to be provided by the caller as input to our function; 3. According to section 7.2.8 ECVP-DSA in [IEEE Std. 1363-:2000].

Application note:

- The certification covers the NIST standard FIPS 186-4 Elliptic Curves P-{192,224,256,384,521} and Brainpool RFC 5639 Elliptic Curves brainpoolIP {192,224,256,320,384,512}r1 and brainpoolIP{192,224,256,320,384,512}t1 and Curve B-{233 283, 409} and Curve K-{233,283,409}.

FCS_COP.1.1/SCP The TSF shall perform **[assignment: session key derivation and data field decryption of the messages exchanged through GlobalPlatform's Secure Channels]** in accordance with a specified cryptographic algorithm **[assignment: TDES in CBC/ECB mode, AES in CBC mode]** and cryptographic key sizes **[assignment: TDES: 112 bits, AES: 128 bits]** that meet the following: **[assignment: TDES: [SP800-67], AES: [FIPS 197], Modes of Operation: [SP800-38A]]**.

FCS_COP.1.1/SCP-AUTH The TSF shall perform **[assignment: authentication cryptogram generation and verification of the messages exchanged through GlobalPlatform's Secure Channels]** in accordance with a specified cryptographic algorithm **[assignment: TDES in CBC mode, KDF in counter mode with C-MAC as PRF]** and cryptographic key sizes **[assignment: TDES: 112 bits, KDF: 128 bits]** that meet the following: **[assignment: TDES: [SP800-67], KDF: [SP800-108], Modes of Operation: [SP800-38A], [SP800-38B]]**.

FCS_COP.1.1/SCP-KA The TSF shall perform **[assignment: session key derivation of the messages exchanged through GlobalPlatform's Secure Channels]** in accordance with a specified cryptographic algorithm **[assignment: Triple-DES in CBC mode with ICV = 0, KDF in counter mode with C-MAC as PRF]** and cryptographic key sizes **[assignment: TDES: 112 bits, KDF: 128 bits]** that meet the following: **[assignment: TDES: [SP800-67], KDF: [SP800-108], Modes of Operation: [SP800-38A], [SP800-38B]]**.

The following two SFRs implement the acceleration of the communication protocol as preparation for an applet that implements secure messaging according to [ICAO Doc 9303] specification.

FCS_COP.1.1/ACC_CYPHER The TSF shall perform **[assignment: secure messaging – encryption and decryption]** in accordance with a specified cryptographic algorithm **[assignment: TDES and AES in CBC mode]** and cryptographic key sizes **[assignment:**

TDES: 112 bits, AES: 128, 192, 256 bits] that meet the following: **[assignment: TDES: [SP800-67], AES: [FIPS 197], Modes of Operation: [SP800-38A]]**.

FCS_COP.1.1/ACC_MAC The TSF shall perform **[assignment: secure messaging – message authentication code]** in accordance with a specified cryptographic algorithm **[assignment: Retail MAC with TDES, AES in C-MAC mode (truncated to 64 bits]** and cryptographic key sizes **[assignment: Retail MAC: 112 bits, CMAC: 128 bits, 192 bits, or 256 bits]** that meet the following: **[assignment: Retail MAC: [ISO9797-1], CMAC: [SP800-38B], TDES: [SP800-67], AES: [FIPS 197]]**

FCS_COP.1.1/DAP/TOKEN The TSF shall perform **[assignment: verification of the DAP signature attached to Executable Load Files, verification of the "Delegated Management Token Signature"[12] attached to card management commands]** in accordance with a specified cryptographic algorithm **[assignment: RSASSA-PSS, RSASSA-PKCS1-v1_5]** and cryptographic key sizes **[assignment: RSA: 1024 - 2048 bits, in increments of 32 bits]** that meet the following: **[assignment: RSA [PKCS V2.1]]**.

Application note:

- RSASSA-PKCS1-v1_5 is supported for 1024-2016 bits, in increments of 32 bits. RSASSA-PSS is supported only for 2048 bits.

FCS_COP.1.1/RECEIPT The TSF shall perform **[assignment: generation of the "Delegated Management Receipt signature" [13] attached to the card management commands responses]** in accordance with a specified cryptographic algorithm **[RSASSA-PKCS1-v1_5,**

[12] A Delegated Management Token Signature is a cryptographic value provided by a Card Issuer as proof that a Delegated Management operation has been authorized

[13] The Delegated Management Receipt Signature is a cryptographic value provided by the card (if so required by the Card Issuer) as proof that a Delegated Management operation has occurred.

DES] and cryptographic key sizes **[assignment:** RSA- **1024 bits,** Retail MAC **- 112 bits]** that meet the following: **[assignment: RSA [PKCS V2.1], Retail MAC [ISO9797-1]]**.

Application note:

- The TOE provides a subset of cryptographic operations defined in [JCAPI3] (see javacardx.crypto.Cipher and javacardx.security packages).
- This component is instantiated according to the version of the Java Card API applicable to this security target and the implemented algorithms ([JCAPI3]).
- The RSA and the ECC modules are optional modules delivery options for the TOE. The TOE can come without the RSA or ECC module. In that case it does not provide the Specific Security Functionality RSA or ECC.

FCS_COP.1.1/PACE_SUPP The TSF shall perform **[assignment: elliptic curve basic operations]** in accordance with a specified cryptographic algorithm **[point addition, scalar multiplication]** and cryptographic key sizes **[assignment: 192, 224, 233, 256, 283, 320, 384, 409, 512, 521 bits]** that meet the following: **[assignment: point addition according to section 2.3.1 in [BSI TR 03111], scalar multiplication according to section 4.3.1 in [BSI TR 03111] without optional Z_AB]**.

Application Note:

- The purpose of this SFR is to support the generic mapping of the PACE protocol, i.e. to provide Javacard api functions for calculating the function GMap() as defined in [BSI TR 03111], chapter 4.4.1
- The certification covers the NIST standard FIPS 186-4 Elliptic Curves P-{192,224,256,384,521} and Brainpool RFC 5639 Elliptic Curves brainpoolIP {192,224,256,320,384,512}r1 and brainpoolIP{192,224,256,320,384,512}t1 and Curve B-{233 283, 409} and Curve K-{233,283,409}.

FCS_COP.1.1/DES-MAC The TSF shall perform **[assignment: MAC calculation and verification]** in accordance with a specified cryptographic algorithm **[TDES in Retail MAC Mode]** and cryptographic key sizes **[assignment: 112 bits]** that meet the following:

[assignment: [SP800-67] with [ISO/IEC9797-1] MAC algorithm 3 / padding method 2].

FCS_COP.1.1/AES-MAC The TSF shall perform **[assignment: MAC calculation and verification]** in accordance with a specified cryptographic algorithm **[AES in CMAC Mode with a MAC length of 16 bytes]** and cryptographic key sizes **[assignment: 128, 192, 256 bits]** that meet the following: **[assignment: [FIPS 197] with [SP 800-38B]**.

FDP_RIP.1/ABORT Subset residual information protection

FDP_RIP.1.1/ABORT The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **any reference to an object instance created during an aborted transaction**.

Application note:

The events that provoke the de-allocation of a transient object are described in [JCRE3], §5.1.

FDP_RIP.1/APDU Subset residual information protection

FDP_RIP.1.1/APDU The TSF shall ensure that any previous information content of a resource is made unavailable upon the **allocation of the resource to** the following objects: **the APDU buffer**.

Application note:

The allocation of a resource to the APDU buffer is typically performed as the result of a call to the process() method of an applet.

FDP_RIP.1/bArray Subset residual information protection

FDP_RIP.1.1/bArray The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **the bArray object**.

Application note:

A resource is allocated to the bArray object when a call to an applet's install() method is performed. There is no conflict with FDP_ROL.1 here because of the bounds on the rollback mechanism (FDP_ROL.1.2/FIREWALL): the scope of the rollback does not extend outside the execution of the install() method, and the de-allocation occurs precisely right after the return of it.

FDP_RIP.1/KEYS Subset residual information protection

FDP_RIP.1.1/KEYS The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **the cryptographic buffer (D.CRYPTO)**.

Application note:

- The javacard.security & javacardx.crypto packages do provide secure interfaces to the cryptographic buffer in a transparent way. See javacard.security.KeyBuilder and Key interface of [JCAPI3].

FDP_RIP.1/TRANSIENT Subset residual information protection

FDP_RIP.1.1/TRANSIENT The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **any transient object**.

Application note:

- The events that provoke the de-allocation of any transient object are described in [JCRE3], §5.1.

- The clearing of CLEAR_ON_DESELECT objects is not necessarily performed when the owner of the objects is deselected. In the presence of multiselectable applet instances, CLEAR_ON_DESELECT memory segments may be attached to applets that are active in different logical channels. Multiselectable applet instances within a same package must share the transient memory segment if they are concurrently active ([JCRE3], §4.2.

FDP_ROL.1/FIREWALL Basic rollback

FDP_ROL.1.1/FIREWALL The TSF shall enforce **the FIREWALL access control SFP and the JCVM information flow control SFP** to permit the rollback of the **operations OP.JAVA and OP.CREATE** on the **object O.JAVAOBJECT**.

FDP_ROL.1.2/FIREWALL The TSF shall permit operations to be rolled back within the **scope of a select(), deselect(), process(), install() or uninstall() call, notwithstanding the restrictions given in [JCRE3], §7.7, within the bounds of the Commit Capacity ([JCRE3], §7.8), and those described in [JCAPI3]**.

Application note:

Transactions are a service offered by the APIs to applets. It is also used by some APIs to guarantee the atomicity of some operation. This mechanism is either implemented in Java Card platform or relies on the transaction mechanism offered by the underlying platform. Some operations of the API are not conditionally updated, as documented in [JCAPI3] (see for instance, PIN-blocking, PIN-checking, update of Transient objects).

6.1.1.3 Card Security Management

FAU_ARP.1 Security alarms

FAU_ARP.1.1 The TSF shall take **one of the following actions:**

- **throw an exception,**
- **lock the card session,**
- **reinitialize the Java Card System and its data,**
- **[assignment: tracking of failures up to the limit after which the card is permanently locked]**

upon detection of a potential security violation.

Refinement:

The "potential security violation" stands for one of the following events:

- CAP file inconsistency,
- typing error in the operands of a bytecode,
- applet life cycle inconsistency,
- card tearing (unexpected removal of the Card out of the CAD) and power failure,
- abort of a transaction in an unexpected context, (see abortTransaction(), [JCAPI3] and ([JCRE3], §7.6.2)
- violation of the Firewall or JCVM SFPs,
- unavailability of resources,
- array overflow,
- stack overflow,
- illegal method arguments
- Integrity checks on critical data structure failures
- Hardware/software countermeasures checking failures

[Confidential information removed in ST Lite]

FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for **[assignment: integrity errors]** on all objects, based on the following attributes: **[assignment: Keys and PIN values integrity security attributes (checksum)]**.

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **[assignment: reset the card]**.

Application note:

- Although no such requirement is mandatory in the Java Card specification, an exception is raised upon integrity errors detection on cryptographic keys, PIN values, sensitive structures and their associated security attributes. Even if all the objects cannot be monitored, cryptographic keys and PIN objects are considered with particular attention by ST authors as they play a key role in the overall security.
- It is also recommended to monitor integrity errors in the code of the native applications and Java Card applets.

- For integrity sensitive application, their data is monitored (D.APP_I_DATA): applications need to protect information against unexpected modifications, and explicitly control whether a piece of information has been changed between two accesses. For example, maintaining the integrity of an electronic purse's balance is extremely important because this value represents real money. Its modification must be controlled, for illegal ones would denote an important failure of the payment system.
- A dedicated library is implemented and made available to developers to achieve better security for specific objects, following the same pattern that already exists in cryptographic APIs, for instance.

FPR_UNO.1 Unobservability

FPR_UNO.1.1 The TSF shall ensure that **all users** are unable to observe the operation **[assignment: End User authentication, cryptographic operation]** on **[assignment: PIN code, TSF data]** by **none.**

Application note:

Although it is not required in [JCRE3] specifications, the non-observability of operations on sensitive information such as keys appears as impossible to circumvent in the smart card world. The precise list of operations and objects is left unspecified, but concern secret keys and PIN codes which exists on the card, as well as the cryptographic operations and comparisons performed on them.

FPT_FLS.1 Failure with preservation of secure state

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur: **those associated to the potential security violations described in FAU_ARP.1**.

Application note:

The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE3], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE3]). Behavior of the TOE on power loss and reset is described in [JCRE3], §3.6 and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE3], §3.6.1.

FPT_TDC.1 Inter-TSF basic TSF data consistency

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret **the CAP files, the bytecode and its data arguments** when shared between the TSF and another trusted IT product.

FPT_TDC.1.2 The TSF shall use

- **the rules defined in [JCVM3] specification,**
- **the API tokens defined in the export files of reference implementation,**
- **[assignment: The ISO 7816-6 rules,**
- **The rules defined in EMV specification,**
- **The rules defined in GP specification]**

when interpreting the TSF data from another trusted IT product.

Application note:

Concerning the interpretation of data between the TOE and the underlying Java Card platform, it is assumed that the TOE is developed consistently with the SCP functions, including memory management, I/O functions and cryptographic functions.

6.1.1.4 AID Management

FIA_ATD.1/AID User attribute definition

FIA_ATD.1.1/AID The TSF shall maintain the following list of security attributes belonging to individual users:

- **Package AID,**
- **Applet's version number,**
- **Registered applet AID,**
- **Applet Selection Status ([JCVM3], §6.5)**.

Refinement:

"Individual users" stand for applets.

FIA_UID.2/AID User identification before any action

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

Application note:

- By users here it must be understood the ones associated to the packages (or applets) that act as subjects of policies. In the Java Card System, every action is always performed by an identified user interpreted here as the currently selected applet or the package that is the subject's owner. Means of identification are provided during the loading procedure of the package and the registration of applet instances.
- The role Java Card RE defined in FMT_SMR.1 is attached to an IT security function rather than to a "user" of the CC terminology. The Java Card RE does not "identify" itself to the TOE, but it is part of it.

FIA_USB.1/AID User-subject binding

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

FIA_USB.1.2/AID The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **[assignment: if an instance of an applet class declared in a certain Java Card package is created, that package is taken as the active context associated to the new application instance.]**.

FIA_USB.1.3/AID The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: **[assignment: if an instance of an applet class declared in a certain Java Card package is created, that package is taken as the active context associated to the new application instance.]**.

Application note:

The user is the applet and the subject is the S.PACKAGE. The subject security attribute "Context" shall hold the user security attribute "package AID".

FMT_MTD.1/JCRE Management of TSF data

FMT_MTD.1.1/JCRE The TSF shall restrict the ability to **modify** the **list of registered applets' AIDs** to **the JCRE**.

Application note:

- The installer and the Java Card RE manage other TSF data such as the applet life cycle or CAP files, but this management is implementation specific. Objects in the Java programming language may also try to query AIDs of installed applets through the lookupAID(...) API method.
- The installer, applet deletion manager or even the card manager may be granted the right to modify the list of registered applets' AIDs in specific implementations (possibly needed for installation and deletion; see #.DELETION and #.INSTALL).

FMT_MTD.3/JCRE Secure TSF data

FMT_MTD.3.1/JCRE The TSF shall ensure that only secure values are accepted for **the registered applets' AIDs**.

6.1.2 InstG Security Functional Requirements

This group consists of the SFRs related to the installation of the applets, which addresses security aspects outside the runtime. The installation of applets is a critical phase, which lies partially out of the boundaries of the firewall, and therefore requires specific treatment. In this ST, loading a package or installing an applet modeled as importation of user data (that is, user application's data) with its security attributes (such as the parameters of the applet used in the firewall rules).

FDP_ITC.2/Installer Import of user data with security attributes

FDP_ITC.2.1/Installer The TSF shall enforce the **PACKAGE LOADING information flow control SFP** when importing user data, controlled under the SFP, from outside of the TOE.

FDP_ITC.2.2/Installer The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3/Installer The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4/Installer The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5/Installer The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE:

Package loading is allowed only if, for each dependent package, its AID attribute is equal to a resident package AID attribute, the major (minor) Version attribute associated to the dependent package is lesser than or equal to the major (minor) Version attribute associated to the resident package ([JCVM3], §4.5.2)..

Application note:

FDP_ITC.2.1/Installer:

- The most common importation of user data is package loading and applet installation on the behalf of the installer. Security attributes consist of the shareable flag of the class component, AID and version numbers of the package, maximal operand stack size and number of local variables for each method, and export and import components (accessibility).

FDP_ITC.2.3/Installer:

- The format of the CAP file is precisely defined in [JCVM3] specifications; it contains the user data (like applet's code and data) and the security attributes altogether. Therefore there is no association to be carried out elsewhere.

FDP_ITC.2.4/Installer:

- Each package contains a package Version attribute, which is a pair of major and minor version numbers ([JCVM3], §4.5). With the AID, it describes the package defined in the CAP file. When an export file is used during preparation of a CAP file, the versions numbers and AIDs indicated in the export file are recorded in the CAP files ([JCVM3], §4.5.2): the dependent packages Versions and AIDs attributes allow the retrieval of these identifications. Checks occur on a case-by-case basis to indicate that package files are binary compatible. However, package files do have "package Version Numbers" ([JCVM3]) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.

FDP_ITC.2.5/Installer:

- A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs.
- The intent of this rule is to ensure the binary compatibility of the package with those already on the card ([JCVM3], §4.4).
- The application instance may only register with the instance AID assigned to it in the INSTALL (install) command by the authenticated client.

FMT_SMR.1/Installer Security roles

FMT_SMR.1.1/Installer The TSF shall maintain the roles**: Installer**.

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

FPT_FLS.1/Installer Failure with preservation of secure state

FPT_FLS.1.1/Installer The TSF shall preserve a secure state when the following types of failures occur: **the installer fails to load/install a package/applet as described in [JCRE3] §11.1.5**.

Application note:

The TOE does not provide additional feedback information to the card manager in case of potential security violations (see FAU_ARP.1). The error status response of [GP] is implemented in the TOE.

FPT_RCV.3/Installer Automated recovery without undue loss

FPT_RCV.3.1/Installer When automated recovery from **[assignment: a failure or a service discontinuity]** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2/Installer For **[assignment: any failure or abortion during the installation process]** the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/Installer The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **[assignment: the entire installed file]** for loss of TSF data or objects under the control of the TSF.

FPT_RCV.3.4/Installer The TSF shall provide the capability to determine the objects that were or were not capable of being recovered.

Application note :

FPT_RCV.3.1/Installer :

- This element is not within the scope of the Java Card specification, which only mandates the behavior of the Java Card System in good working order. The following is an excerpt from [CC2], p298: In this maintenance mode normal operation might be impossible or severely restricted, as otherwise insecure situations might occur. Typically, only authorized users should be allowed access to this mode but the real details of who can access this mode is a function of FMT: Security management. If FMT: Security management does not put any controls on who can access this mode, then it may be acceptable to allow any user to restore the system if the TOE enters such a state. However, in practice, this is probably not desirable as the user restoring the system has an opportunity to configure the TOE in such a way as to violate the SFRs.

FPT_RCV.3.2/Installer:

- Should the installer fail during loading/installation of a package/applet, it has to revert to a "consistent and secure state". The Java Card RE has some clean up duties as well; see [JCRE3], §11.1.6 for possible scenarios. This component includes among the listed failures the deletion of a package/applet. In the case of a failure during loading/installation of a package/applet, all allocations are undone (by unrolling the transaction). The original state is restored. The install/load operation can be retried after correcting the original failure reason.

- Other events such as the unexpected tearing of the card, power loss, and so on, are partially handled by the underlying hardware platform (see [PP0035]) and, from the TOE's side, by events "that clear transient objects" and transactional features. See FPT_FLS.1.1, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT and FDP_ROL.1/FIREWALL.

FPT_RCV.3.3/Installer:

- First, the SCP ensures the atomicity of updates for fields and objects, and a power-failure during a transaction or the normal runtime does not create the loss of otherwise-permanent data, in the sense that memory on a smart card is essentially persistent with this respect (Flash). Data stored on the RAM and subject to such failure is intended to have a limited lifetime anyway (runtime data on the stack, transient objects' contents). According to this, the loss of data within the TSF scope is limited to the same restrictions of the transaction mechanism.

6.1.3 ADELG Security Functional Requirements

This group consists of the SFRs related to the deletion of applets and/or packages, enforcing the applet deletion manager (ADEL) policy on security aspects outside the runtime. Deletion is a critical operation and therefore requires specific treatment. This policy is better thought as a frame to be filled by ST implementers.

FDP_ACC.2/ADEL Complete access control

FDP_ACC.2.1/ADEL The TSF shall enforce the **ADEL access control SFP** on **S.ADEL, S.JCRE, S.JCVM, O.JAVAOBJECT, O.APPLET and O.CODE_PKG** and all operations among subjects and objects covered by the SFP.

Refinement:

The operations involved in the policy are:

- OP.DELETE_APPLET,
- OP.DELETE_PCKG,
- OP.DELETE_PCKG_APPLET.

FDP_ACC.2.2/ADEL The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.

FDP_ACF.1/ADEL Security attribute based access control

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

Subject/Object	Attributes
S.JCVM	Active Applets
S.JCRE	Selected Applet Context, Registered Applets, Resident Packages
O.CODE_PKG	Package AID, Dependent Package AID, Static References
O.APPLET	Applet Selection Status
O.JAVAOBJECT	Owner

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

In the context of this policy, an object O is reachable if and only one of the following conditions hold:

- **(1) the owner of O is a registered applet instance A (O is reachable from A),**
- **(2) a static field of a resident package P contains a reference to O (O is reachable from P),**
- **(3) there exists a valid remote reference to O (O is remote reachable),**
- **(4) there exists an object O' that is reachable according to either (1) or (2) or (3) above and O' contains a reference to O (the reachability status of O is that of O').**

The following access control rules determine when an operation among controlled subjects and objects is allowed by the policy:

- **R.JAVA.14 ([JCRE3], §11.3.4.1, Applet Instance Deletion): S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if,**

(1) S.ADEL is currently selected,

(2) there is no instance in the context of O.APPLET that is active in any logical channel and

(3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE3], §8.5) O.JAVAOBJECT is remote reachable.

- **R.JAVA.15 ([JCRE3], §11.3.4.1, Multiple Applet Instance Deletion): S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if,**

(1) S.ADEL is currently selected,

(2) there is no instance of any of the O.APPLET being deleted that is active in any logical channel and

(3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE3], §8.5) O.JAVAOBJECT is remote reachable.

- **R.JAVA.16 ([JCRE3], §11.3.4.2, Applet/Library Package Deletion): S.ADEL may perform OP.DELETE_PCKG upon an O.CODE_PKG only if,**

(1) S.ADEL is currently selected,

(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PKG that is an instance of a class that belongs to O.CODE_PKG, exists on the card and

(3) there is no resident package on the card that depends on O.CODE_PKG.

- **R.JAVA.17 ([JCRE3], §11.3.4.3, Applet Package and Contained Instances Deletion): S.ADEL may perform OP.DELETE_PCKG_APPLET upon an O.CODE_PKG only if,**

(1) S.ADEL is currently selected,

(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PKG, which is an instance of a class that belongs to O.CODE_PKG exists on the card,

(3) there is no package loaded on the card that depends on O.CODE_PKG, and

(4) for every O.APPLET of those being deleted it holds that: (i) there is no instance in the context of O.APPLET that is active in any logical channel and (ii) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted, or ([JCRE3], §8.5) O.JAVAOBJECT is remote reachable.

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

FDP_ACF.1.4/ADEL [Editorially Refined] The TSF shall explicitly deny access of **any subject but S.ADEL to O.CODE_PKG or O.APPLET for the purpose of deleting them from the card**.

Application note:

FDP_ACF.1.2/ADEL:

- This policy introduces the notion of reachability, which provides a general means to describe objects that are referenced from a certain applet instance or package.
- S.ADEL calls the "uninstall" method of the applet instance to be deleted, if implemented by the applet, to inform it of the deletion request. The order in which these calls and the dependencies checks are performed are out of the scope of this protection profile.

FDP_RIP.1/ADEL Subset residual information protection

FDP_RIP.1.1/ADEL The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **applet instances and/or packages when one of the deletion operations in FDP_ACC.2.1/ADEL is performed on them**.

Application note:

Deleted freed resources (both code and data) may be reused, depending on the way they were deleted (logically or physically). Requirements on de-allocation during applet/package deletion are described in [JCRE3], §11.3.4.1, §11.3.4.2 and §11.3.4.3.

FMT_MSA.1/ADEL Management of security attributes

FMT_MSA.1.1/ADEL The TSF shall enforce the **ADEL access control SFP** to restrict the ability to **modify** the security attributes **Registered Applets and Resident Packages** to **the Java Card RE**.

FMT_MSA.3/ADEL Static attribute initialization

FMT_MSA.3.1/ADEL The TSF shall enforce the **ADEL access control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/ADEL The TSF shall allow the **following role(s): none,** to specify alternative initial values to override the default values when an object or information is created.

FMT_SMF.1/ADEL Specification of Management Functions

FMT_SMF.1.1/ADEL The TSF shall be capable of performing the following management functions: **modify the list of registered applets' AIDs and the Resident Packages.**

FMT_SMR.1/ADEL Security roles

FMT_SMR.1.1/ADEL The TSF shall maintain the roles**: applet deletion manager**.

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

FPT_FLS.1/ADEL Failure with preservation of secure state

FPT_FLS.1.1/ADEL The TSF shall preserve a secure state when the following types of failures occur: **the applet deletion manager fails to delete a package/applet as described in [JCRE3], §11.3.4**.

Application note:

- The TOE provides additional feedback information to the card manager in case of a potential security violation (see FAU_ARP.1).
- The Package/applet instance deletion must be atomic. The "secure state" referred to in the requirement complies with Java Card specification ([JCRE3], §11.3.4.). If the Card Manager is unable to delete all objects of the applet instance being deleted, it will abort the deletion attempt and return a failure code. Similarly, package deletion is atomic. This behavior is conformant to the Java Card specification. No special tracking of application/package

deletion failures is performed. But, failures in secure channel creation induce velocity checking countermeasures and slow down reattempts.

6.1.4 ODELG Security Functional Requirements

The following requirements concern the object deletion mechanism. This mechanism is triggered by the applet that owns the deleted objects by invoking a specific API method.

FDP_RIP.1/ODEL Subset residual information protection

FDP_RIP.1.1/ODEL The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **the objects owned by the context of an applet instance which triggered the execution of the method `javacard.framework.JCSystem.requestObjectDeletion()`.**

Application note:

- Freed data resources resulting from the invocation of the method javacard.framework.JCSystem.requestObjectDeletion() may be reused. Requirements on de-allocation after the invocation of the method are described in [JCAPI3].
- There is no conflict with FDP_ROL.1 here because of the bounds on the rollback mechanism: the execution of requestObjectDeletion() is not in the scope of the rollback because it must be performed in between APDU command processing, and therefore no transaction can be in progress.

FPT_FLS.1/ODEL Failure with preservation of secure state

FPT_FLS.1.1/ODEL The TSF shall preserve a secure state when the following types of failures occur: **the object deletion functions fail to delete all the unreferenced objects owned by the applet that requested the execution of the method**.

Application note:

The TOE may provide additional feedback information to the card manager in case of potential security violation (see FAU_ARP.1). If there is a failure when attempting to delete the unreferenced objects, these objects continue to be remain unreferenced and continue to remain as garbage. The security state is not perturbed. Since the "object deletion" function was not completed, it is automatically retried on the next available opportunity (i.e. between APDUs).

6.1.5 Security Functional Requirements

This group includes requirements for preventing the installation of packages that has not been bytecode verified, or that has been modified after bytecode verification.

FCO_NRO.2/CM Enforced proof of origin

FCO_NRO.2.1/CM The TSF shall enforce the generation of evidence of origin for transmitted **[assignment: application package, the Security Domain with Delegated Management privilege]**at all times.

FCO_NRO.2.2/CM [Editorially Refined] The TSF shall be able to relate the **identity** of the originator of the information, and the **[assignment: application package contained in, the request command parameters of]**the information to which the evidence applies.

FCO_NRO.2.3/CM The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **[assignment: by the DAP verification, the Token verification].**

Application note:

FCO_NRO.2.1/CM:

- Upon reception of a new application package for installation, the card manager first checks that it actually comes from the verification authority. The verification authority is the entity responsible for bytecode verification.

FCO_NRO.2.3/CM:

- The rules described in GP 2.2 are enforced. Package download via INSTALL (load) command must be performed over a secure channel preceded by client authentication. Integrity checking of the package payload is performed.

FDP_IFC.2/CM Complete information flow control

FDP_IFC.2.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** on **S.INSTALLER, S.BCV, S.CAD and I.APDU** and all operations that cause that information to flow to and from subjects covered by the SFP.

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

Application note:

- The subjects covered by this policy are those involved in the loading of an application package by the card through a potentially unsafe communication channel.
- The operations that make information to flow between the subjects are those enabling to send a message through and to receive a message from the communication channel linking the card to the outside world. It is assumed that any message sent through the channel as clear text can be read by an attacker. Moreover, an attacker may capture any message sent through the communication channel and send its own messages to the other subjects.
- The information controlled by the policy is the APDUs exchanged by the subjects through the communication channel linking the card and the CAD. Each of those messages contain part of an application package that is required to be loaded on the card, as well as any control information used by the subjects in the communication protocol.

FDP_IFF.1/CM Simple security attributes

FDP_IFF.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** based on the following types of subject and information security attributes: **[assignment: keys used for DAP]**:

Subjects	Security attributes
DAP Keys used by S.INSTALLER	Valid key

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

rules describing the communication protocol used by the CAD and the card for transmitting a new package.

- **The subject S.INSTALLER shall accept a message only if it comes from the subject S.CAD.**
- **The subject S.INSTALLER shall accept an application package only if it has received without modification and in the right order all the APDUs sent by the subject S.CAD]**

FDP_IFF.1.3/CM The TSF shall enforce the **[assignment: additional information flow control SFP rules: none]**.

FDP_IFF.1.4/CM The TSF shall explicitly authorize an information flow based on the following rules: **[assignment: Rules defined in [GP] §E and [GP_ID], §3]**.

FDP_IFF.1.5/CM The TSF shall explicitly deny an information flow based on the following rules:

- **The TOE fails to verify the integrity and authenticity evidences of the application package**
- **[assignment: The retry counter limit is exceeded].**

Application note:

FDP_IFF.1.1/CM:

- The security attributes used to enforce the PACKAGE LOADING SFP depend on the communication protocol enforced between the CAD and the card. For instance, some of the attributes that can be used are: (1) the keys used by the subjects to encrypt/decrypt their messages; (2) the number of pieces the application package has been split into in order to be sent to the card; (3) the ordinal of each piece in the decomposition of the package, etc. See for example Appendix D of [GP].

FDP_IFF.1.2/CM:

- The whole exchange of messages verify at least the following two rules: (1) the subject S.INSTALLER shall accept a message only if it comes from the subject S.CAD; (2) the subject S.INSTALLER shall accept an application package only if it has received without modification and in the right order all the APDUs sent by the subject S.CAD.

FDP_IFF.1.5/CM:

- The verification of the integrity and authenticity evidences is performed either during loading or during the first installation of an application of the package.

FDP_UIT.1/CM Data exchange integrity

FDP_UIT.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to **[selection: transmit, receive]** user data in a manner protected from **[selection: modification, deletion, insertion, replay]** errors.

FDP_UIT.1.2/CM [Editorially Refined] The TSF shall be able to determine on receipt of user data, whether **modification, deletion, insertion, replay of some of the pieces of the application sent by the CAD** has occurred.

Application note:

Modification errors are understood as modification, substitution, unrecoverable ordering change of data and any other integrity error that may cause the application package to be installed on the card to be different from the one sent by the CAD. The TOE implements [GP]

FIA_UID.1/CM Timing of identification

FIA_UID.1.1/CM The TSF shall allow **[assignment: application selection; initializing a secure channel with the card; requesting data that identifies the card or the Card Issuer]** on behalf of the user to be performed before the user is identified.

FIA_UID.1.2/CM The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

Application note:

The package installation requires the user to be identified. Here by user is meant the one(s) that in the Security Target shall be associated to the role(s) defined in the component FMT_SMR.1/CM.

FMT_MSA.1/CM Management of security attributes

FMT_MSA.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to restrict the ability to **[selection: modify, delete, [assignment: reset]]** the security attributes **[assignment: keys used for DAP]** to **[assignment: S.INSTALLER]**.

FMT_MSA.3/CM Static attribute initialization

FMT_MSA.3.1/CM The TSF shall enforce the **[assignment: PACKAGE LOADING information flow control SFP]** to provide **[selection: restrictive]** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/CM The TSF shall allow the **[assignment: the authorized identified roles: none]** to specify alternative initial values to override the default values when an object or information is created.

FMT_SMF.1/CM Specification of Management Functions

FMT_SMF.1.1/CM The TSF shall be capable of performing the following management functions: **[assignment: modify the security attributes].**

FMT_SMR.1/CM Security roles

FMT_SMR.1.1/CM The TSF shall maintain the roles: **[assignment: Card Manager]**.

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

FTP_ITC.1/CM Inter-TSF trusted channel

FTP_ITC.1.1/CM The TSF shall provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC.1.2/CM [Editorially Refined] The TSF shall permit **the CAD placed in the card issuer secured environment** to initiate communication via the trusted channel.

FTP_ITC.1.3/CM The TSF shall initiate communication via the trusted channel for **loading/installing a new application package on the card**.

Application note:

There is no dynamic package loading on the Java Card platform. New packages can be installed on the card only on demand of the card issuer. The TOE implements SCP02 and SCP03 with the following options. SCP02: 0x15, 0x1A, and 0x55. SCP03 Option 0x00, 0x10

6.1.6 EMG Security Functional Requirements

The group EMG contains the following security requirements for the management of the external memory, introduced in the version 2.2.2 of the Java Card System (cf. [JCAPI3], optional package javacardx.external). These SFRs cover the OS Extension Architecture mechanism as defined in Section 1.4.4 which allows extension of the OS without the need to recompile and deploy a new JCS image.

FDP_ACC.1/EXT_MEM Subset access control

FDP_ACC.1.1/EXT_MEM The TSF shall enforce the **EXTERNAL MEMORY access control SFP** on **subject S.APPLET, object O.EXT_MEM_INSTANCE, and operations OP.CREATE_EXT_MEM_INSTANCE, OP.READ_EXT_MEM and OP.WRITE_EXT_MEM.**

FDP_ACF.1/EXT_MEM Security attribute based access control

FDP_ACF.1.1/EXT_MEM The TSF shall enforce the **[assignment: EXTERNAL MEMORY access control SFP]** to objects based on the following: **[assignment:]**

Object	Security attribute
O.EXT_MEM_INSTANCE	Address space.

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

- R.JAVA.20: Any subject S.APPLET that performs OP.CREATE_EXT_MEM_INSTANCE obtains an object O.EXT_MEM_INSTANCE that addresses a memory space different from that of the Java Card System.
- R.JAVA.21: Any subject S.APPLET may perform OP.READ_EXT_MEM (O.EXT_MEM_INSTANCE, address) provided the address belongs to the space of the O.EXT_MEM_INSTANCE.
- R.JAVA.22: Any subject S.APPLET may perform OP.WRITE_EXT_MEM (O.EXT_MEM_INSTANCE, address) provided the address belongs to the space of the O.EXT_MEM_INSTANCE.

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

FDP_ACF.1.4/EXT_MEM The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[assignment: none]**.

Application note:

This rule only states that the accessible address space must not interfere with that of the Java Card System.

The creation and the access to an external memory instance fall in the scope of the Firewall rules.

FMT_MSA.1/EXT_MEM Management of security attributes

FMT_MSA.1.1/EXT_MEM The TSF shall enforce the **EXTERNAL MEMORY access control SFP** to restrict the ability to **set up** the security attributes **address space** to **the Java Card RE**.

FMT_MSA.3/EXT_MEM Static attribute initialization

FMT_MSA.3.1/EXT_MEM The TSF shall enforce the **EXTERNAL MEMORY access control SFP** to provide **no** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/EXT_MEM The TSF shall allow the **Java Card RE** to specify alternative initial values to override the default values when an object or information is created.

Application note:

Upon creation of an external memory instance, the Java Card RE gets the address space value for the newly created object.

FMT_SMF.1/EXT_MEM Specification of Management Functions

FMT_SMF.1.1/EXT_MEM The TSF shall be capable of performing the following management functions: **set up the address space security attribute**.

6.1.7 GPG Security Functional Requirements

The following SFRs are related to the security requirements for the GlobalPlatform card manager functionalities of the TOE.

FDP_ACC.1/GPG Subset access control

FDP_ACC.1.1/GPG The TSF shall enforce the **[assignment: CARD CONTENT MANAGEMENT access control SFP]** on [**assignment: Subjects: S.INSTALLER, S.PACKAGE, S.ADEL, S.CAD, Objects: O.APPLET, O.CODE_PKG; Operations: OP.GP]**

FDP_ACF.1/GPG Security attribute based access control

FDP_ACF.1.1/GPG The TSF shall enforce the **[assignment: CARD CONTENT MANAGEMENT access control SFP]** to objects based on the following: **[assignment:]**

Subject/Object	Security attributes
S.INSTALLER	None
S.PACKAGE	LC Selection Status, Currently Active Context
S.ADEL	None
S.CAD	None
O.APPLET, O.CODE_PKG	None

FDP_ACF.1.2/GPG The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **[assignment: CARD CONTENT MANAGEMENT access control SFP: OP.GP(...)]**

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

FDP_ACF.1.4/GPG The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[assignment: If the card life cycle state is TERMINATED or**

LOCKED the access of subjects for CARD CONTENT MANAGEMENT access control SFP on its objects is not allowed.]

FMT_MSA.1/GPG Management of security attributes

FMT_MSA.1.1/GPG The TSF shall enforce the **[assignment: CARD CONTENT MANAGEMENT access control SFP]** to restrict the ability to **[selection: modify]** the security attributes**: [assignment: Card Life Cycle State, Security Level]** to **[assignment: S.INSTALLER].**

FMT_MSA.3/GPG Static attribute initialization

FMT_MSA.3.1/GPG The TSF shall enforce the **[assignment: CARD CONTENT MANAGEMENT access control SFP]** to provide **[selection: restrictive]** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/GPG The TSF shall allow the **[assignment: the authorized identified roles: none]** to specify alternative initial values to override the default values when an object or information is created.

FMT_SMF.1/GPG Specification of Management Functions

FMT_SMF.1.1/GPG The TSF shall be capable of performing the following management functions: **[assignment: Modification of the security attributes Card Life Cycle State and Security Level]**.

FMT_SMR.1/GPG Security roles

FMT_SMR.1.1/GPG The TSF shall maintain the roles **[assignment: S.CAD, and S.INSTALLER]**.

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

FIA_UID.1/GPG Timing of identification

FIA_UID.1.1/GPG The TSF shall allow **[assignment: TSF-mediated actions: GET DATA, INITIALIZE UPDATE, EXTERNAL AUTHENTICATE according to [GP]]** on behalf of the user to be performed before the user is identified.

FIA_UID.1.2/GPG The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

Application Note:

The list of TSF-mediated actions is implementation-dependent, but package installation requires the user to be identified. Here by user is meant the one(s) that in the Security Target shall be associated to the role(s) defined in the component FMT_SMR.1/GPG.

6.1.8 SCPG Security Functional Requirements

The group SCPG contains the security requirements, which are needed to support the platform.

FCS_RNG.1 Quality metric for Random Numbers

The TOE meets the requirement “Quality metric for random numbers (FCS_RNG.1)” as specified below (Common Criteria Part 2 extended).

FCS_RNG.1.1 The TSF shall provide a [**selection: deterministic**] random number generator that implements:

- **(DRG.4.1)** The internal state of the RNG uses a PTRNG of class PTG.2 as a random source.
- **(DRG.4.2)** The RNG provides forward secrecy.
- **(DRG.4.3)** The RNG provides backward secrecy, even if the current internal state is known.
- **(DRG.4.4)** The RNG provides enhanced forward secrecy **[assignment: for every call]**.
- **(DRG.4.5)** The internal state of the RNG is seeded by **[selection: a PTRNG of class PTG.2]**.

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

- **(DRG.4.6)** The RNG generates output for which **[2^{35}]** strings of bit length 128 are mutually different with probability **[assignment: less than 2^{-16}]**.
- **(DRG.4.7)** Statistical test suites cannot practically distinguish the random number from output sequences of an ideal RNG. The random numbers pass test procedure A **[assignment: defined in AIS20]**.

FPT_EMSEC.1 TOE Emanation

FPT_EMSEC.1.1 The TOE shall not emit **[assignment: variations in power consumption or timing during command execution]** enabling access to **[assignment: TSF data: D.JCS_KEYs and D.CRYPTO]** and **[assignment: User data: D.PIN, D.APP_KEYs]**.

FPT_EMSEC.1.2 The TSF shall ensure **[assignment: that any users]** are unable to use the following interface **[assignment: circuit interface]** to gain access to **[assignment: TSF data: D.JCS_KEYs and D.CRYPTO]** and **[assignment: User data: D.PIN, D.APP_KEYs]**.

FPT_RCV.3/SCP Automated recovery without undue loss

FPT_RCV.3.1/SCP When automated recovery from **[assignment: loss of power and card tearing]** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2/SCP For **[assignment: loss of power and card tearing]** the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/SCP The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **[assignment 0%]** for loss of TSF data or objects within the TSC.

FPT_RCV.3.4/SCP The TSF shall provide the capability to determine the objects that were or were not capable of being recovered.

FPT_RCV.4/SCP Function recovery

FPT_RCV.4.1/SCP The TSF shall ensure that **[assignment: reading from and writing to static and objects' fields interrupted by power loss]** have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

6.2 TOE Security Assurance Requirements

The security assurance requirement level is EAL5 augmented by AVA_VAN.5 "Advanced methodical vulnerability analysis" and ALC_DVS.2 "Sufficiency of security measures".

6.3 Security Requirements Rationale

6.3.1 Objectives

6.3.1.1 Security Objectives for the TOE

6.3.1.1.1 IDENTIFICATION

O.SID Subjects' identity is AID-based (applets, packages), and is met by the following SFRs: FDP_ITC.2/Installer, FIA_ATD.1/AID, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_MSA.1/REM_REFS, FMT_MSA.1/ADEL, FMT_MSA.1/CM, FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.3/CM, FMT_SMF.1/CM, FMT_SMF.1/ADEL, FMT_SMF.1/ADEL, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_SMF.1/EXT_MEM, FMT_MSA.1/EXT_MEM and FMT_MSA.3/EXT_MEM.

6.3.1.1.2 EXECUTION

O.FIREWALL This objective is met by the FIREWALL access control policy FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL, the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) and the functional requirement FDP_ITC.2/Installer. The functional requirements of the class FMT (FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMF.1, FMT_SMR.1/ADEL, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_SMF.1/EXT_MEM, FMT_MSA.1/EXT_MEM, FMT_MSA.3/EXT_MEM, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_MSA.2/FIREWALL_JCVM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM) also indirectly contribute to meet this objective.

O.GLOBAL_ARRAYS_CONFID Only arrays can be designated as global, and the only global arrays required in the Java Card API are the APDU buffer and the global byte array input parameter (bArray) to an applet's install method. The clearing requirement of these arrays is met by (FDP_RIP.1/APDU and FDP_RIP.1/bArray respectively). The JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) prevents an application from keeping a pointer to a shared buffer, which could be used to read its contents when the buffer is being used by another application.

Protection of the array parameters of remotely invoked methods, which are global as well, is covered by the general initialization of method parameters (FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL and FDP_RIP.1/TRANSIENT).

O.GLOBAL_ARRAYS_INTEG This objective is met by the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM), which prevents an application from keeping a pointer to the APDU buffer of the card or to the global byte array of the applet's install method. Such a pointer could be used to access and modify it when the buffer is being used by another application.

O.NATIVE This security objective is covered by FDP_ACF.1/FIREWALL: the only means to execute native code is the invocation of a Java Card API method. This objective mainly relies on the environmental objective OE.APPLET, which uphold the assumption A.APPLET.

O.OPERATE The TOE is protected in various ways against applets' actions (FPT_TDC.1), the FIREWALL access control policy FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL, and is able to detect and block various failures or security violations during usual working (FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FAU_ARP.1). Its security-critical parts and procedures are also protected: safe recovery from failure is ensured (FPT_RCV.3/Installer), applets' installation may be cleanly aborted (FDP_ROL.1/FIREWALL), communication with external users and their internal subjects is well-controlled (FDP_ITC.2/Installer, FIA_ATD.1/AID, FIA_USB.1/AID) to prevent alteration of TSF data (also protected by components of the FPT class).

Almost every objective and/or functional requirement indirectly contributes to this one too.

Application note: Startup of the TOE (TSF-testing) can be covered by FPT_TST.1. This SFR component is not mandatory in [JCRE3], but appears in most of security requirements documents for masked applications. Self tests are performed on startup.

O.REALLOCATION This security objective is satisfied by the following SFRs: FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/ADEL, which imposes that the contents of the re-allocated block shall always be cleared before delivering the block.

O.RESOURCES The TSFs detects stack/memory overflows during execution of applications (FAU_ARP.1, FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer). Failed installations are not to create memory leaks (FDP_ROL.1/FIREWALL, FPT_RCV.3/Installer) as well. Memory management is controlled by the TSF (FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMF.1 FMT_SMR.1/ADEL, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_SMF.1/EXT_MEM and FMT_SMR.1/CM).

6.3.1.1.3 SERVICES

O.ALARM This security objective is met by FPT_FLS.1/Installer, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL which guarantee that a secure state is preserved by the TSF when failures occur, and FAU_ARP.1 which defines TSF reaction upon detection of a potential security violation.

O.CIPHER This security objective is directly covered by FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4 and FCS_COP.1. The SFR FPR_UNO.1 contributes in covering this security objective and controls the observation of the cryptographic operations which may be used to disclose the keys.

O.KEY-MNGT This relies on the same security functional requirements as O.CIPHER, plus FDP_RIP.1 and FDP_SDI.2 as well. Precisely it is met by the following components: FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FCS_COP.1, FPR_UNO.1, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL and FDP_RIP.1/TRANSIENT.

O.PIN-MNGT This security objective is ensured by FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FPR_UNO.1, FDP_ROL.1/FIREWALL and FDP_SDI.2 security functional requirements. The TSFs behind these are implemented by API classes. The firewall security functions FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL shall protect the access to private and internal data of the objects.

O.TRANSACTION Directly met by FDP_ROL.1/FIREWALL, FDP_RIP.1/ABORT, FDP_RIP.1/ODEL, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT and FDP_RIP.1/OBJECTS (more precisely, by the element FDP_RIP.1.1/ABORT).

6.3.1.1.4 OBJECT DELETION

O.OBJ-DELETION This security objective specifies that deletion of objects is secure. The security objective is met by the security functional requirements FDP_RIP.1/ODEL and FPT_FLS.1/ODEL.

6.3.1.1.5 APPLET MANAGEMENT

O.DELETION This security objective specifies that applet and package deletion must be secure. The non-introduction of security holes is ensured by the ADEL access control policy (FDP_ACC.2/ADEL, FDP_ACF.1/ADEL). The integrity and confidentiality of data that does not belong to the deleted applet or package is a by-product of this policy as well. Non-accessibility of deleted data is met by FDP_RIP.1/ADEL and the TSFs are protected against possible failures of the deletion procedures (FPT_FLS.1/ADEL, FPT_RCV.3/Installer). The security functional requirements of the class FMT (FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL) included in the group ADELG also contribute to meet this objective.

O.LOAD This security objective specifies that the loading of a package into the card must be secure. Evidence of the origin of the package is enforced (FCO_NRO.2/CM) and the integrity of the corresponding data is under the control of the PACKAGE LOADING information flow policy (FDP_IFC.2/CM, FDP_IFF.1/CM) and FDP_UIT.1/CM. Appropriate identification (FIA_UID.1/CM) and transmission mechanisms are also enforced (FTP_ITC.1/CM).

O.INSTALL This security objective specifies that installation of applets must be secure. Security attributes of installed data are under the control of the FIREWALL access control policy (FDP_ITC.2/Installer), and the TSFs are protected against possible failures of the installer (FPT_FLS.1/Installer, FPT_RCV.3/Installer).

*6.3.1.1.6 **Communication***

O.COMMUNICATION This security objective is met by FMT_SMR.1/GPG, it specifies the authorized identified roles enabling to send and authenticate card management commands. FMT_MSA.1/GPG and FMT_MSA.3/GPG cover indirectly this security objective by specifying security attributes enabling to guarantee the integrity of card management requests. FIA_UID.1/GPG specify the actions that can be performed prior to identification of the origin of the APDU commands that the TOE receives

O.RECOVERY This objective is met by the component FPT_RCV.3/SCP. The components FPT_RCV.3 and FPT_RCV.4 are used to support the objective O.OS_SUPPORT and O. RECOVERY to assist the TOE to recover in the event of a power failure. FAU_ARP.1 reacts to the detection of a potential security violation, while FPT_FLS.1 preserves a secure state. If the power fails or the card is withdrawn prematurely from the CAD the operation of the TOE may be interrupted leaving the TOE in an inconsistent state

*6.3.1.1.7 **Management of the external memory***

O.EXT-MEM The Java Card System memory is protected against applet's attempts of unauthorized access through the external memory facilities by the EXTERNAL MEMORY access control policy (FDP_ACC.1/EXT_MEM, FDP_ACF.1/EXT_MEM), which first controls the accessible address space, then controls the effective read and write operations. External memory management is controlled by the TSF (FMT_SMF.1/EXT_MEM).

6.3.1.1.8 Random Numbers

O.RND This objective is met by the component FCS_RNG.1, as [AIS20] needs non-predictable random numbers with the needed cryptographic quality.

6.3.1.1.9 Card Manager

O.CARD-MANAGEMENT The objective is met by the security requirements for the GlobalPlatform card manager functionalities of the TOE. It protects against unauthorized access of the CARD CONTENT MANAGEMENT access control SFP (FDP_ACC.1/GPG and FDP_ACF.1/GPG). The GP management is controlled by the TSF (FMT_MSA.1/GPG, FMT_SMF.1/GPG, FMT_MSA.3.1/GPG, FMT_SMR.1/GPG, FIA_UID.1/GPG).

O.OS_SUPPORT The objective is met by the components low-level cryptographic support by FCS_COP.1, low-level transaction support by FDP_ROL.1/FIREWALL, low-level data integrity monitoring by FDP_SDI.2, the automated recovery of FPT_RCV.3/SCP and FPT_RCV.4/SCP.

O.IC_SUPPORT The objective is met by the components that protect against physical attacks, i.e. the strong random number generator supported by the platform true random number generator FCS_RNG, not emit variations in power consumption or timing during command execution by FPT_EMSEC

6.4 Rationale tables of Security Objectives and SFRs

Security Objectives	Security Functional Requirements	Rationale
O.SID	FIA_ATD.1/AID, FMT_MSA.1/JCRE, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.1/CM, FMT_MSA.3/CM, FDP_ITC.2/Installer, FMT_SMF.1/CM, FMT_SMF.1/ADEL, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_MSA.1/JCVM, FMT_MSA.3/JCVM, FMT_SMF.1/EXT_MEM, FMT_MSA.1/EXT_MEM, FMT_MSA.3/EXT_MEM	Section 6.3.1.1.1

O.FIREWALL	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FMT_SMR.1/Installer, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_MSA.3/FIREWALL, FMT_SMR.1, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FMT_MSA.1/JCRE, FDP_ITC.2/Installer, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_SMF.1, FMT_MSA.2/FIREWALL_JCVM, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_MSA.1/JCVM, FMT_MSA.3/JCVM FMT_SMF.1/EXT_MEM, FMT_MSA.1/EXT_MEM, FMT_MSA.3/EXT_MEM	Section 6.3.1.1.2
O.GLOBAL_ARRAYS_CONFID	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FDP_RIP.1/bArray, FDP_RIP.1/APDU, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT	Section 6.3.1.1.2
O.GLOBAL_ARRAYS_INTEG	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM	Section 6.3.1.1.2
O.NATIVE	FDP_ACF.1/FIREWALL	Section 6.3.1.1.2
O.OPERATE	FAU_ARP.1, FDP_ROL.1/FIREWALL, FIA_ATD.1/AID, FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FDP_ITC.2/Installer, FPT_RCV.3/Installer, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FPT_TDC.1, FIA_USB.1/AID	Section 6.3.1.1.2
O.REALLOCATION	FDP_RIP.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS	Section 6.3.1.1.2
O.RESOURCES	FAU_ARP.1, FDP_ROL.1/FIREWALL, FMT_SMR.1/Installer, FMT_SMR.1,	Section 6.3.1.1.2

	FMT_SMR.1/ADEL, FPT_FLS.1/Installer, FPT_FLS.1/ODEL, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_RCV.3/Installer, FMT_SMR.1/CM, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_SMF.1, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE FMT_SMF.1/EXT_MEM	
O.ALARM	FPT_FLS.1/Installer, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL, FAU_ARP.1	Section 6.3.1.1.3
O.CIPHER	FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FCS_COP.1, FCS_RNG.1, FPR_UNO.1	Section 6.3.1.1.3
O.KEY-MNGT	FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FCS_COP.1, FPR_UNO.1, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_SDI.2, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT	Section 6.3.1.1.3
O.PIN-MNGT	FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FPR_UNO.1, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FDP_ROL.1/FIREWALL, FDP_SDI.2, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL	Section 6.3.1.1.3
O.TRANSACTION	FDP_ROL.1/FIREWALL, FDP_RIP.1/ABORT, FDP_RIP.1/ODEL, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FDP_RIP.1/OBJECTS	Section 6.3.1.1.3
O.OBJ-DELETION	FDP_RIP.1/ODEL, FPT_FLS.1/ODEL	Section 6.3.1.1.4
O.DELETION	FDP_ACC.2/ADEL, FDP_ACF.1/ADEL, FDP_RIP.1/ADEL, FPT_FLS.1/ADEL,	Section 6.3.1.1.5

	FPT_RCV.3/Installer, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL	
O.LOAD	FCO_NRO.2/CM, FDP_IFC.2/CM, FDP_IFF.1/CM, FDP_UIT.1/CM, FIA_UID.1/CM, FTP_ITC.1/CM	Section 6.3.1.1.5
O.INSTALL	FDP_ITC.2/Installer, FPT_RCV.3/Installer, FPT_FLS.1/Installer	Section 6.3.1.1.5
O.COMMUNICATION	FMT_SMR.1/GPG, FMT_MSA.1/GPG, FMT_MSA.3/GPG, FIA_UID.1/GPG	Section 6.3.1.1.6
O.RECOVERY	FPT_FLS.1, FPT_FLS.1/SCP, FPT_RCV.3/SCP, FPT_RCV.4/SCP, FAU_ARP.1	Section 6.3.1.1.6
O.EXT-MEM	FDP_ACC.1/EXT_MEM, FDP_ACF.1/EXT_MEM, FMT_SMF.1/EXT_MEM	Section 6.3.1.1.7
O.RND	FCS_RNG.1	Section 6.3.1.1.8
O.CARD-MANAGEMENT	FDP_ACC.1/GPG, FDP_ACF.1/GPG, FMT_MSA.1/GPG, FMT_SMF.1/GPG, FMT_MSA.3/GPG, FMT_SMR.1/GPG, FIA_UID.1/GPG	Section 6.3.1.1.9
O.OS_SUPPORT	FCS_COP.1, FDP_ROL.1/FIREWALL, FDP_SDI.2, FPT_RCV.3/SCP, FPT_RCV.4/SCP	Section 6.3.1.1.9
O.IC_SUPPORT	FPT_EMSEC.1, FCS_RNG.1 (Hardware Platform SFR),	Section 6.3.1.1.9

Table 14 Security Objectives and SFRs – Coverage

Security Functional Requirements	Security Objectives
FDP_ACC.2/FIREWALL	O.FIREWALL, O.OPERATE, O.PIN-MNGT
FDP_ACF.1/FIREWALL	O.FIREWALL, O.NATIVE, O.OPERATE, O.PIN-MNGT
FDP_IFC.1/JCVM	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG
FDP_IFF.1/JCVM	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG
FDP_RIP.1/OBJECTS	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FMT_MSA.1/JCRE	O.SID, O.FIREWALL
FMT_MSA.1/JCVM	O.SID, O.FIREWALL
FMT_MSA.2/FIREWALL_JCVM	O.FIREWALL
FMT_MSA.3/FIREWALL	O.SID, O.FIREWALL
FMT_MSA.3/JCVM	O.SID, O.FIREWALL
FMT_SMF.1	O.FIREWALL, O.RESOURCES
FMT_SMR.1	O.FIREWALL, O.RESOURCES
FCS_CKM.1	O.CIPHER, O.KEY-MNGT
FCS_CKM.2	O.CIPHER, O.KEY-MNGT
FCS_CKM.3	O.CIPHER, O.KEY-MNGT
FCS_CKM.4	O.CIPHER, O.KEY-MNGT
FCS_COP.1	O.CIPHER, O.KEY-MNGT
FDP_RIP.1/ABORT	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FDP_RIP.1/APDU	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FDP_RIP.1/bArray	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FDP_RIP.1/KEYS	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION

Security Functional Requirements	Security Objectives
FDP_RIP.1/TRANSIENT	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FDP_ROL.1/FIREWALL	O.OPERATE, O.RESOURCES, O.PIN-MNGT, O.TRANSACTION
FAU_ARP.1	O.OPERATE, O.RESOURCES, O.ALARM
FDP_SDI.2	O.KEY-MNGT, O.PIN-MNGT
FPR_UNO.1	O.CIPHER, O.KEY-MNGT, O.PIN-MNGT
FPT_FLS.1	O.OPERATE, O.RESOURCES, O.ALARM
FPT_TDC.1	O.OPERATE
FIA_ATD.1/AID	O.SID, O.OPERATE
FIA_USB.1/AID	O.SID, O.OPERATE
FMT_MTD.1/JCRE	O.SID, O.FIREWALL, O.RESOURCES
FMT_MTD.3/JCRE	O.SID, O.FIREWALL, O.RESOURCES
FDP_ITC.2/Installer	O.SID, O.FIREWALL, O.OPERATE, O.INSTALL
FMT_SMR.1/Installer	O.FIREWALL, O.RESOURCES
FPT_FLS.1/Installer	O.OPERATE, O.RESOURCES, O.ALARM, O.INSTALL
FPT_RCV.3/Installer	O.OPERATE, O.RESOURCES, O.DELETION, O.INSTALL, O.RECOVERY
FDP_ACC.2/ADEL	O.DELETION
FDP_ACF.1/ADEL	O.DELETION
FDP_RIP.1/ADEL	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.DELETION, O.REALLOCATION
FMT_MSA.1/ADEL	O.SID, O.FIREWALL, O.DELETION
FMT_MSA.3/ADEL	O.SID, O.FIREWALL, O.DELETION
FMT_SMF.1/ADEL	O.SID, O.FIREWALL, O.RESOURCES
FMT_SMR.1/ADEL	O.FIREWALL, O.RESOURCES, O.DELETION
FPT_FLS.1/ADEL	O.OPERATE, O.RESOURCES, O.ALARM, O.DELETION
FDP_RIP.1/ODEL	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.OBJ-DELETION, O.REALLOCATION

Security Functional Requirements	Security Objectives
FPT_FLS.1/ODEL	O.OPERATE, O.RESOURCES, O.ALARM, O.OBJ-DELETION
FCO_NRO.2/CM	O.LOAD
FDP_IFC.2/CM	O.LOAD
FDP_IFF.1/CM	O.LOAD
FDP_UIT.1/CM	O.LOAD
FIA_UID.1/CM	O.LOAD
FMT_MSA.1/CM	O.SID, O.FIREWALL
FMT_MSA.3/CM	O.SID, O.FIREWALL
FMT_SMF.1/CM	O.SID, O.FIREWALL, O.RESOURCES
FMT_SMR.1/CM	O.FIREWALL, O.RESOURCES
FTP_ITC.1/CM	O.LOAD
FDP_ACC.1/EXT_MEM	O.EXT-MEM
FDP_ACF.1/EXT_MEM	O.EXT-MEM
FMT_MSA.1/EXT_MEM	O.SID, O.FIREWALL
FMT_MSA.3/EXT_MEM	O.SID, O.FIREWALL
FMT_SMF.1/EXT_MEM	O.SID, O.RESOURCES, O.FIREWALL, O.EXT-MEM
FDP_ACC.1/GPG	O.CARD-MANAGEMENT
FDP_ACF.1/GPG	O.CARD-MANAGEMENT
FMT_MSA.1/GPG	O.COMMUNICATION, O.CARD-MANAGEMENT
FMT_SMF.1/GPG	O.CARD-MANAGEMENT
FMT_MSA.3/GPG	O.COMMUNICATION, O.CARD-MANAGEMENT
FMT_SMR.1/GPG	O.COMMUNICATION, O.CARD-MANAGEMENT
FIA_UID.1/GPG	O.COMMUNICATION, O.CARD-MANAGEMENT
FCS_RNG.1	O.RND
FPT_EMSEC.1	O.IC_SUPPORT
FPT_RCV.3/SCP	O.RECOVERY, O.OS_SUPPORT
FPT_RCV.4/SCP	O.RECOVERY, O.OS_SUPPORT

Table 15 SFRs and Security Objectives

6.5 Dependencies

6.5.1 SFRs dependencies

Requirements	CC Dependencies	Satisfied Dependencies
FDP_ITC.2/Installer	(FDP_ACC.1 or FDP_IFC.1) and (FPT_TDC.1) and (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM, FPT_TDC.1
FMT_SMR.1/Installer	(FIA_UID.1)	
FPT_FLS.1/Installer	No dependencies	
FPT_RCV.3/Installer	(AGD_OPE.1)	AGD_OPE.1
FDP_ACC.2/ADEL	(FDP_ACF.1)	FDP_ACF.1/ADEL
FDP_ACF.1/ADEL	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.2/ADEL, FMT_MSA.3/ADEL
FDP_RIP.1/ADEL	No dependencies	
FMT_MSA.1/ADEL	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FMT_MSA.1/ADEL, FMT_SMR.1/ADEL
FMT_MSA.3/ADEL	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/ADEL, FMT_SMR.1/ADEL
FMT_SMF.1/ADEL	No dependencies	
FMT_SMR.1/ADEL	(FIA_UID.1)	
FPT_FLS.1/ADEL	No dependencies	
FDP_RIP.1/ODEL	No dependencies	
FPT_FLS.1/ODEL	No dependencies	
FCO_NRO.2/CM	(FIA_UID.1)	FIA_UID.1/CM
FDP_IFC.2/CM	(FDP_IFF.1)	FDP_IFF.1/CM

FDP_IFF.1/CM	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/CM, FMT_MSA.3/CM
FDP_UIT.1/CM	(FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM
FIA_UID.1/CM	No dependencies	
FMT_MSA.1/CM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_IFC.2/CM, FMT_SMF.1/CM, FMT_SMR.1/CM
FMT_MSA.3/CM	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/CM, FMT_SMR.1/CM
FMT_SMF.1/CM	No dependencies	
FMT_SMR.1/CM	(FIA_UID.1)	FIA_UID.1/CM
FTP_ITC.1/CM	No dependencies	
FDP_ACC.2/FIREWALL	(FDP_ACF.1)	FDP_ACF.1/FIREWALL
FDP_ACF.1/FIREWALL	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.2/FIREWALL, FMT_MSA.3/FIREWALL
FDP_IFC.1/JCVM	(FDP_IFF.1)	FDP_IFF.1/JCVM
FDP_IFF.1/JCVM	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.1/JCVM, FMT_MSA.3/JCVM
FDP_RIP.1/OBJECTS	No dependencies	
FMT_MSA.1/JCRE	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FMT_SMR.1
FMT_MSA.1/JCVM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_SMF.1, FMT_SMR.1

FMT_MSA.2/FIREWALL_JCVM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1
FMT_MSA.3/FIREWALL	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1
FMT_MSA.3/JCVM	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/JCVM, FMT_SMR.1
FMT_SMF.1	No dependencies	
FMT_SMR.1	(FIA_UID.1)	FIA_UID.2/AID
FCS_CKM.1	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_CKM.2, FCS_CKM.4
FCS_CKM.2	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1, FCS_CKM.4
FCS_CKM.3	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1, FCS_CKM.4
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	FCS_CKM.1
FCS_COP.1	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1, FCS_CKM.4
FDP_RIP.1/ABORT	No dependencies	
FDP_RIP.1/APDU	No dependencies	
FDP_RIP.1/bArray	No dependencies	
FDP_RIP.1/KEYS	No dependencies	
FDP_RIP.1/TRANSIENT	No dependencies	

FDP_ROL.1/FIREWALL	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM
FAU_ARP.1	(FAU_SAA.1)	
FDP_SDI.2	No dependencies	
FPR_UNO.1	No dependencies	
FPT_FLS.1	No dependencies	
FPT_TDC.1	No dependencies	
FIA_ATD.1/AID	No dependencies	
FIA_UID.2/AID	No dependencies	
FIA_USB.1/AID	(FIA_ATD.1)	FIA_ATD.1/AID
FMT_MTD.1/JCRE	(FMT_SMF.1) and (FMT_SMR.1)	FMT_SMF.1, FMT_SMR.1
FMT_MTD.3/JCRE	(FMT_MTD.1)	FMT_MTD.1/JCRE
FDP_ACC.1/EXT_MEM	(FDP_ACF.1)	FDP_ACF.1/EXT_MEM
FDP_ACF.1/EXT_MEM	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/EXT_MEM, FMT_MSA.3/EXT_MEM
FMT_MSA.1/EXT_MEM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.1/EXT_MEM, FMT_SMF.1/EXT_MEM
FMT_MSA.3/EXT_MEM	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/EXT_MEM
FMT_SMF.1/EXT_MEM	No dependencies	
FDP_ACC.1/GPG	(FDP_ACF.1)	FDP_ACF.1/GPG
FDP_ACF.1/GPG	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/GPG, FMT_MSA.3/GPG

FMT_MSA.1/GPG	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.1/GPG, FMT_SMF.1/GPG, FMT_SMR.1/GPG
FMT_SMF.1/GPG	No dependencies	
FMT_MSA.3/GPG	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/GPG, FMT_SMR.1/GPG
FMT_SMR.1/GPG	(FIA_UID.1)	FIA_UID.1/GPG
FIA_UID.1/GPG	No dependencies	
FCS_RNG.1	No dependencies	
FPT_EMSEC.1	No dependencies	
FPT_RCV.3/SCP	(AGD_OPE.1)	AGD_OPE.1
FPT_RCV.4/SCP	No dependencies	

Table 16 SFRs dependencies

6.5.2 Rationale for the exclusion of dependencies

The dependency FIA_UID.1 of FMT_SMR.1/Installer is unsupported. This ST does not require the identification of the "installer" since it can be considered as part of the TSF.

The dependency FIA_UID.1 of FMT_SMR.1/ADEL is unsupported. This ST does not require the identification of the "deletion manager" since it can be considered as part of the TSF.

The dependency FMT_SMF.1 of FMT_MSA.1/JCRE is unsupported. The dependency between FMT_MSA.1/JCRE and FMT_SMF.1 is not satisfied because no management functions are required for the Java Card RE.

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. On the contrary, the events listed in FAU_ARP.1 are self-contained (arithmetic exception, ill-formed bytecodes, access failure) and ask for a straightforward reaction of the TSFs on their occurrence at runtime. The JCVM or other components of the TOE detect these events during their usual working order. Thus, there is no mandatory audit recording in this ST.

The dependency FCS_CKM.2 on FCS_CKM.1 is only partially supported. The dependency of FCS_CKM.2 on FCS_CKM.1 is supported only for asymmetric cryptography. The TOE does not provide SFRs for symmetric key (DES and AES) generation and therefore there is no dependency.

The dependency FCS_CKM.3 on FCS_CKM.1 is only partially supported. The dependency of FCS_CKM.3 on FCS_CKM.1 is supported only for asymmetric cryptography. The TOE does not provide SFRs for symmetric (DES and AES) key generation and therefore there is no dependency.

6.5.3 SARs dependencies

Requirements	CC Dependencies	Satisfied Dependencies
ADV_ARC.1	(ADV_FSP.1) and (ADV_TDS.1)	ADV_FSP.4, ADV_TDS.3
ADV_FSP.5	(ADV_TDS.1) and (ADV_IMP.1)	ADV_TDS.3, ADV_IMP.1
ADV_IMP.1	(ADV_TDS.3) and (ALC_TAT.1)	ADV_TDS.3, ALC_TAT.2
ADV_INT.2	(ADV_IMP.1) and (ADV_TDS.3) and (ALC_TAT.1)	ADV_TDS.3, ADV_IMP.1, ALC_TAT.2
ADV_TDS.4	(ADV_FSP.5)	ADV_FSP.5
AGD_OPE.1	(ADV_FSP.1)	ADV_FSP.5
AGD_PRE.1	No dependencies	
ALC_CMC.4	(ALC_CMS.1) and (ALC_DVS.1) and (ALC_LCD.1)	ALC_CMS.4, ALC_DVS.2, ALC_LCD.1
ALC_CMS.5	No dependencies	
ALC_DEL.1	No dependencies	
ALC_DVS.2	No dependencies	
ALC_LCD.1	No dependencies	
ALC_TAT.2	(ADV_IMP.1)	ADV_IMP.1

ASE_CCL.1	(ASE_ECD.1) and (ASE_INT.1) and (ASE_REQ.1)	ASE_ECD.1, ASE_INT.1, ASE_REQ.2
ASE_ECD.1	No dependencies	
ASE_INT.1	No dependencies	
ASE_OBJ.2	(ASE_SPD.1)	ASE_SPD.1
ASE_REQ.2	(ASE_ECD.1) and (ASE_OBJ.2)	ASE_ECD.1, ASE_OBJ.2
ASE_SPD.1	No dependencies	
ASE_TSS.1	(ADV_FSP.1) and (ASE_INT.1) and (ASE_REQ.1)	ADV_FSP.5, ASE_INT.1, ASE_REQ.2
ATE_COV.2	(ADV_FSP.2) and (ATE_FUN.1)	ADV_FSP.5, ATE_FUN.1
ATE_DPT.3	(ADV_ARC.1) and (ADV_TDS.2) and (ATE_FUN.1)	ADV_ARC.1, ADV_TDS.3, ATE_FUN.1
ATE_FUN.1	(ATE_COV.1)	ATE_COV.2
ATE_IND.2	(ADV_FSP.2) and (AGD_OPE.1) and (AGD_PRE.1) and (ATE_COV.1) and (ATE_FUN.1)	ADV_FSP.5, AGD_OPE.1, AGD_PRE.1, ATE_COV.2, ATE_FUN.1
AVA_VAN.5	(ADV_ARC.1) and (ADV_FSP.4) and (ADV_IMP.1) and (ADV_TDS.3) and (AGD_OPE.1) and (AGD_PRE.1) and (ATE_DPT.1)	ADV_ARC.1, ADV_FSP.5, ADV_IMP.1, ADV_TDS.3, AGD_OPE.1, AGD_PRE.1, ATE_DPT.1

Table 17 SARs dependencies

6.5.4 Rationale for the Security Assurance Requirement

EAL5 is chosen for this TOE and product since it is intended to defend against highly sophisticated attacks. This evaluation assurance level allows a developer to gain maximum assurance from positive security engineering based on good practices. EAL5 represents the highest practical level of assurance expected for a commercial grade product. In order to provide a meaningful level of assurance that the TOE and its embedding product provide an adequate level of defense against such attacks: the evaluators should have access to the semiformal modular TSF design. The lowest EAL level for which such access is provided is EAL5.

6.5.4.1 ALC_DVS.2 Sufficiency of security measures

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 and embedding product, it is necessary to justify the sufficiency of these procedures to protect their confidentiality and integrity. ALC_DVS.2 has no dependencies.

6.5.4.2 AVA_VAN.5 Advanced methodical vulnerability analysis

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 in payment and identity areas. AVA_VAN.5 has dependencies on ADV_ARC.1, ADV_FSP.1, ADV_TDS.3, ADV_IMP.1, AGD_PRE.1 and AGD_OPE.1. All of them are satisfied by EAL5.

6.6 Extended Components Definition

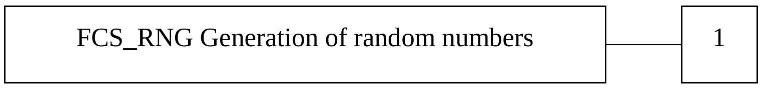
6.6.1 Definition of the Family FCS_RNG

This chapter has been copied from the certified (BSI-PP-0035) Smartcard IC Platform Protection Profile [PP0035].

Family behavior

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

Component leveling:



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 actions defined to be auditable.
FCS_RNG.1	Quality metric for random numbers
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.

6.6.2 Definition of the Family FPT_EMSEC

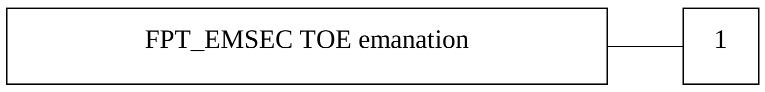
This chapter has been copied from the certified Protection Profile [PP0017] and [PP0035].

The additional family FPT_EMSEC (TOE Emanation) of the Class FPT (Protection of the TSF) is defined here to describe the IT security functional requirements of the TOE. The TOE shall prevent attacks against the private signature key and other secret data where the attack is based on external observable physical phenomena of the TOE. Examples of such attacks are evaluation of TOE's electromagnetic radiation, simple power analysis (SPA), differential power analysis (DPA), timing attacks, etc. This family describes the functional requirements for the limitation of intelligible emanations which are not directly addressed by any other component of Common Criteria part 2.

Family behaviour

This family defines requirements to mitigate intelligible emanations.

Component behaviour:



FPT_EMSEC.1 TOE emanation has two constituents:

FPT_EMSEC.1.1	Limit of emissions requires to not emit intelligible emissions enabling access to TSF data or user data.
FPT_EMSEC.1.2	Interface emanation requires not emit interface emanation enabling access to TSF data or user data.
Management:	FPT_EMSEC.1

There are no management activities foreseen.

Audit:	FPT_EMSEC.1

There are no actions defined to be auditable.

FPT_EMSEC.1	TOE Emanation
Hierarchical to:	No other components.
FPT_EMSEC.1.1	The TOE shall not emit [assignment: types of emissions] in excess of [assignment: specified limits] enabling access to [assignment: list of types of TSF data] and [assignment: list of types of user data].
FPT_EMSEC.1.2	The TSF shall ensure [assignment: type of users] are unable to use the following interface [assignment: type of connection] to gain access to [assignment: list of types of TSF data] and [assignment: list of types of user data].
Dependencies:	No other components.

CHAPTER

7 TOE Summary Specification

This chapter summarizes the security functionality and the assurance measures of the TOE that fulfill the TOE security requirements. The next table lists the summary of the security functionality of the TOE.

[Confidential information removed in ST Lite]- This applies to the whole Chapter.

TOE Security Functionality	Mapping to SFR
GLOBALPLATFORM’s TOE Security Functionality	
SF.Card Manager	FAU_ARP.1 FIA_ATD.1/AID FIA_UID.2/AID FIA_UID.1/CM FCO_NRO.2/CM FDP_ACC.1/GPG FDP_ACF.1/GPG FDP_ACC.2/ADEL FDP_IFF.1/CM FDP_ITC.2/Installer FDP_UIT.1/CM FTP_ITC.1/CM FIA_UID.1/CM FMT_MSA.1/CM FMT_MSA.1/JCVM FMT_MSA.1/ADEL FMT_MSA.3/ADEL FMT_MSA.3/CM

	FMT_MSA.3/JCVM FMT_MTD.1/JCRE FMT_MTD.3/JCRE FMT_SMF.1 FMT_SMF.1/ADEL FMT_SMF.1/CM FMT_SMF.1/GPG FMT_SMR.1/ADEL FMT_SMR.1/CM FMT_SMR.1/Installer FPT_FLS.1 FPT_FLS.1/ADEL FPT_FLS.1/Installer FPT_FLS.1/ODEL FPT_RCV.3/Installer FPT_TDC.1
SF.Secure Channels	FDP_RIP.1/TRANSIENT FDP_ACF.1/ADEL FDP_ACF.1/FIREWALL FAU_ARP.1 FPR_UNO.1, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL FDP_RIP.1/TRANSIENT
SF.Secure Channel Key Management	FDP_RIP.1 FDP_SDI.2 FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FCS_COP.1
SF.Global PIN Management	FDP_RIP.1/ODEL,

	FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FPR_UNO.1, FDP_ROL.1/FIREWALL FDP_SDI.2 FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
Java Card Runtime Environment TOE Security Functionality	
SF.Java Card Firewall	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL FAU_ARP.1 FMT_SMR.1/GPG FDP_IFC.1/JCVM FDP_IFC.2/CM FDP_IFF.1/JCVM FIA_USB.1/AID FMT_MSA.1/GPG FMT_MSA.2/FIREWALL_JCVM FMT_MSA.3/FIREWALLFMT_MSA.3/GPG FIA_UID.1/GPG FMT_MSA.1/JCRE
SF.End User Authentication	FMT_SMR.1 FPR_UNO.1
SF.Sensitive Data Cleaner	FDP_RIP.1/OBJECTS FDP_RIP.1/ABORT FDP_RIP.1/APDU FDP_RIP.1/bArray FDP_RIP.1/KEYS FDP_RIP.1/TRANSIENT FDP_RIP.1/ADEL FDP_RIP.1/ODEL

SF.Atomic_Transactions	FDP_ROL.1/FIREWALL FPT_RCV.3/SCP FPT_RCV.4/SCP
SF.Security Violation	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
SF.PIN integrity	FAU_ARP.1 FPR_UNO.1 FDP_SDI.2
SF.Key Management	FAU_ARP.1 FCS_COP.1 FCS_CKM.1 FCS_CKM.2 FCS_CKM.3 FCS_CKM.4 FDP_SDI.2
SF.Cryptographic Operations	FAU_ARP.1 FCS_COP.1 FCS_CKM.1 FCS_CKM.2 FCS_CKM.3 FCS_CKM.4 FCS_RNG.1 FPT_EMSEC.1
SF.Extended Memory	FDP_ACC.1/EXT_MEM FDP_ACF.1/EXT_MEM FMT_MSA.1/EXT_MEM FMT_MSA.3/EXT_MEM FMT_SMF.1/EXT_MEM

Table 18 SFRs dependencies

7.1 Security Functionality

7.1.1 GLOBALPLATFORM’s TOE Security Functionality

SF.Card Manager

In Open Mode configuration, the Card Manager is activated and is responsible for card administration. The goal of the Card Manager is to enforce the security policies of the Card Issuer on the card by providing the following features:

Card Content Management (CCM)

This feature is the capability for loading, installing, register updating and removal of card content (executable load file).

DAP Verification

This feature guarantees the authenticity of the Executable Load File by verifying the DAP signature on the Load File using the Verification Authority's RSA public key.

This is an optional feature of the Java Card Platform and will be activated only if the card has been configured to do so. If the TOE has been configured to verify the DAP then the Verification Authority signs the Application electronically (loading file) right after its verification according to [APPNOTE]. Otherwise if the TOE has not been configured to verify DAP signatures, the Verification Authority shall transmit application to the IC Manufacturer/Composite Product Integrator/Card Issuer through a secure communication channel ensuring the origin and the integrity of transmitted files. Upon reception, the IC Manufacturer/Composite Product Integrator/Card Issuer shall store the application in its secure environment until the file is downloaded into the card.

Card Management Environment

This feature initializes and manages the GlobalPlatform Registry data such as the currently loaded applets instances and AIDs or the Security Domain associations and privileges.

APDU Commands Dispatcher

This TOE Security Functionality is in charge of dispatching the APDU commands received by the Card Manager to the selected applet installed on the card. It is the responsibility of the application to correctly reject the command if necessary. Note that also the Proprietary APDUs do go through the Card Manager and its dispatcher.

SSD Delegated Management

This feature enables Delegated Card Content Management which allows the Application Provider's Security Domain to perform delegated loading, delegated installation and make selectable, delegated extradition, delegated update of GP registry and delegated deletion.

This feature is realized by the Advanced GP module which includes Delegated Management with token verification and receipt generation, Install for Registry Update and Install for Extradition.

This is an optional feature of the Java Card Platform and will be activated only if the card has been configured to do so

Life-Cycle Management

This feature controls the state transition of the card and the installed applications as required by the GlobalPlatform specification [GP]. It ensures that initialization and personalization dedicated commands are disabled in the usage stage of the TOE to prevent any unauthorized modification of the TSF data.

Logical Channel Management

This feature enables the multi-selection of applet instance.

SF.Secure Channels

This TOE Security Functionality provides a secure mean for the *IC Manufacturer/Composite Product Integrator/Card Issuer* to perform card management. This TSF protects the sensitive assets exchanged during that process. It relies on the Secure Channel Protocols defined in GlobalPlatform specification [GP]. This is achieved by the following features:

Mutual Authentication

The *IC Manufacturer/Composite Product Integrator/Card Issuer* is authenticated through the following Secure Channel Protocols compliant with the GP specification:

SCP02 (with options 0x15, 0x1a, or 0x55)

Or, SCP03 (with options 0x00 or 0x10)

Depending on the configuration of the Card, only one of the two protocols must be activated to execute mutual authentication operation.

Message Integrity Verification

This feature ensures the integrity of the APDU commands received through a Secure Channel.

Message Confidentiality

This feature decrypts the contents of APDU commands received through a Secure Channel, guaranteeing the confidentiality of the sensitive assets.

Secure Messaging acceleration

This feature allows the applet developer to implement fast secure transfer of data. (Native implementation of secure messaging protocol according to ICAO specification is provided to the applet developer through proprietary interfaces.). This feature is realized by the LDS Module (LDS Secure Messaging Accelerator Module).

This is an optional feature of Java Card Platform and will be activated only if the card has been configured to do so.

SF.Secure Channel Key Management

This TOE security Functionality is intended to securely manage the keys used to establish a secure channel. These are the Session keys used to open a secure channel with the CAD and the ISD keys used to open a secure channel with the IC Manufacturer/Composite Product Integrator/Card Issuer.

Session Key/ISD Key Generation

The TOE generates the keys in accordance with the Java Card Specification and supported by the platform true random number generator.

SF.Global PIN Management

This TOE Security Functionality controls the update of the security attributes associated the global CVM which is restricted to the applets installed with the CVM Privilege.

7.1.2 Java Card TOE Security Functionality

SF.Java Card Firewall

The Java Card firewall provides protection against the most frequently anticipated security concern: developer mistakes and design oversights that might allow sensitive data to be “leaked” to another applet. However, if the object is owned by an applet protected by its own firewall, the requesting applet must satisfy certain access rules before it can use the reference to access the object.

This TSF is enforced by the Java Card Virtual Machine. Each applet has its own bytecode, set of APDUs and classes. The instantiated applet gains the capability to have exclusive ownership of data object constructed from its classes. When an applet is activated, it is given private, exclusive ownership of a logical communication channel. In addition, it gains a unique execution context (instruction counter, status and stack).

The firewall also provides protection against incorrect code. If incorrect code is loaded onto a card, the firewall still protects objects from being accessed by this code.

SF.End User Authentication

This TOE Security Functionality allows and applet's user identification and authentication using the following features:

PIN comparison feature

This feature is based on a secure comparison between the PIN value stored in the persistent memory and the value received from the CAD.

SF.Sensitive Data Cleaner

This TOE Security Functionality ensures that sensitive information contained in data containers (APDU buffer, cryptographic buffer, local variables, bArray, static fields, Class instances fields, etc.) are cleared after usage upon sensitive operations (Deletion of Packages/Applets/Objects, Cryptographic operations, APDU commands, etc.).

SF.Atomic_Transactions

This TOE Security Functionality ensures the atomicity of transactions. It manages the contents of persistent storage after a stop, failure, or fatal exception during an update of a single object field or single class field or single array component. An applet might need to atomically update several different fields or array components in several different objects. Either all updates take place correctly and consistently, or else all fields or components are restored to their previous values

SF.Security Violation

This TSF detects an attempt to illegally access an object belonging to another applet across the firewall boundary, on violation of fundamental language restrictions, such as attempting to invoke a private method in another class, on unavailability of data upon allocation.

SF.PIN integrity

This TOE Security Functionality ensures that the PIN value is protected in integrity. The integrity value is checked as well as its persistent attributes before any operation made on the PIN value.

SF.Key Management

This TOE Security Functionality ensures a secure on-card cryptographic keys infrastructure. Thus, providing the following security features:

Keys Integrity Protection:

This feature guarantees that the cryptographic keys are protected from unauthorized modification.

Keys Confidentiality Protection:

This feature guarantees that the cryptographic keys are protected from disclosure.

Keys Secure Generation:

This feature provides applets with on-card generation of cryptographic keys.

Keys Secure Deletion:

This feature ensures that the keys can never be disclosed after deletion. Therefore, upon deletion, the content of the key objects is erased and the allocated memory space is free again.

Keys Secure Distribution:

This feature enforces the distribution of the cryptographic keys.

Keys Secure Agreement:

This feature ensures a secure sharing of secrets between the applet and the terminal. For that, the applet instances use key agreement algorithms.

SF.Cryptographic Operations

This TOE Security Functionality enforces security means to execute the following cryptographic operations:

Message Digest Generation

This feature generates a hash value from a block of data contained in a byte array. It is almost impossible to generate one hash value from two different blocks of data.

Signature Generation & Verification

This feature is responsible for generating and verifying an electronic signature. It uses a private key to generate the electronic signature contained in a byte array and a public key to verify the signature. In addition it uses a hash function in the signature generation process to obtain a condensed version of the data to be signed/ a message digest. The message digest is given in input to the digital signature algorithm to generate the digital signature.

Encryption & Decryption

This feature is responsible for encrypting and decrypting the contents of a byte array.

Unique Hash Value

This feature enables the generation of a unique hash values (SHA-1 or SHA-2) for application instance's data.

ECC basic operations

This feature enables elliptic curve basic operations (point addition, scalar multiplication).

MAC calculation and verification

This feature enable the calculation and verification of MACs in accordance with TDES in Retail Mac Mode and AES in CMAC Mode.

Random Number Generation

This feature enables the generation of random data to be used for generating cryptographic protocol challenges and cryptographic key values.

SF.Extended Memory

This feature provides controlled access means to the external memory and ensures that the external memory does not address Java Card System memory (containing User Data and TSF Data) and the extended JC/GP functionality does not interfere with the TOEs memory.

7.2 Statement of Compatibility Concerning Composite Security Target

This is a composite evaluation. For this kind of evaluation the BSI defines the [AIS36] composite evaluation scheme. The compatibility between this Composite Security Target (this ST) and the Platform Security Target [ST_IC] is claimed.

In the section "Usage of Platform TSF by TOE TSF", the separation of the relevant security functionality described in the ST of the smart card platform used by the TOE.

The Java Card Protection Profile defines miscellaneous sets of possible Java Card Systems as a TOE. See Appendix 1 of [PP] for details.

The [PP] defines Groups, which can be chosen for different evaluations. For this Security Target the following groups are part of the TOE.

The TOE is the Java Card System Open 3 Classic Edition with External memory group (EMG) added and without the RMI group (RMIG). So the following groups are part of the TOE:

- Core with Logical Channels (CoreG_LC)
- Installer (InstG)
- Object deletion (ODELG)
- Applet deletion (ADELG)
- Secure carrier (CarG)

- External memory (EMG)
- GlobalPlatform (GPG), this is an addition to the Java Card PP

The smart card platform (SCP) is composed of the Infineon micro-controller and hardware abstraction layer containing the cryptographic library.

Apart from Java Card applications, the final product may contain native applications as well.

Group	Description	TOE
Core (CoreG_LC)	The CoreG_LC contains the requirements concerning the runtime environment of the Java Card System implementing logical channels. This includes the firewall policy and the requirements related to the Java Card API.	TOE
Installer *(InstG)*	The InstG contains the security requirements concerning the installation of post-issuance applications. It does not address card management issues in the broad sense, but only those security aspects of the installation procedure that are related to applet execution.	TOE
RMI (RMIG)	The RMIG contains the security requirements for the remote method invocation feature. This is an optional group in PP. This group is not used in this ST.	Non TOE
Object deletion *(ODELG)*	The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism.	TOE
Applet deletion *(ADELG)*	The ADELG contains the security requirements for erasing installed applets from the card.	TOE
Secure carrier *(CarG)*	The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing, in those configurations that do not support on-card static or dynamic bytecode verification, the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.	TOE

External memory *(EMG)*	The EMG contains the requirements for a secure management of the external memory accessible to applet instances.	TOE
GlobalPlatform *(GPG)*	The Card Manager is conformant to the GlobalPlatform Card Specification 2.2 [GP].	TOE
Smart card platform *(SCPG)*	The SCPG contains the security requirements for the smart card platform, that is, native operating system and chip that the Java Card System is implemented upon. In this Security Target, this group applies to the composite platform and is within the scope of evaluation.	TOE

The following sections further describe the separation of the relevant security functionality of the smart card platform, i.e. the crypto library of the M7892 G12, which is needed by the TOE.

7.3 Assumptions and OSP of the Platform for its Operational Environment

The assumptions A.APPLET and A.VERIFICATION include no assumption for the platform. They are related to the processes and users of the TOE.

There is no significant platform assumption (SgPA) of the Platform-ST and no conflict to the security environments of the Platform-ST [ST_IC].

Assumptions of the hardware platform [ST_IC] chapter 4.3	Description	Categorization, see [AIS36]	Remark
A.Process-Sec-IC	Protection during Packaging, Finishing and Personalization	Fulfilled (CfPA)	Fulfilled by the composite-SAR class ALC
A.Resp-Appl	Treatment of User Data	Fulfilled (CfPA)	The TOE implements the needed functionality as A.Resp-Appl can be mapped to O.KEY-MNGT and O.PIN-MNGT
A.Key-Function	Usage of Key-dependent Functions	Fulfilled (CfPA)	The TOE implements the needed functionality as A.Key-Function can be mapped to T.PHYSICAL and T.CONFID-APPLI-DATA

There are three OSPs mentioned in the ST of the hardware platform [ST_IC], P.Process-TOE , P.Add-Functions and P.Crypto-Service.

OSP of the hardware platform [ST_IC] chapter 3.3	Description	Applicable threats by TOE	Remark
P.Process-TOE	Protection during TOE Development and Production	T.CONFID-JCS-CODE T.CONFID-JCS-DATA T.INTEG-APPLI-CODE T.INTEG-APPLI-CODE.LOAD T.INTEG-APPLI-DATA T.INTEG-APPLI-DATA.LOAD T.INTEG-JCS-CODE T.INTEG-JCS-DATA	The OSP of the hardware platform is not contradictory to the threats of the Composite-ST
P.Add-Functions	specific security functionality to the TOE	T.CONFID-APPLI-DATA T.CONFID-JCS-DATA T.INTEG-APPLI-DATA T.INTEG-JCS-DATA T.SID.2 T.RESOURCES	matches O.IC_SUPPORT
P.Crypto-Service	Cryptographic services of the TOE	T.CONFID-APPLI-DATA T.CONFID-JCS-DATA T.INTEG-APPLI-DATA T.INTEG-JCS-DATA T.SID.2 T.RESOURCES	matches O.IC_SUPPORT

7.4 Threats of the Platform

The threats T.PHYSICAL, T.FAULT, T.LEAKAGE, and T.RND can be directly mapped to the platform threats, as listed in the following table. The other threats of the TOE are not related to the platform IC.

Platform threats (Hardware and Crypto Library)	TOE threats	Remark
T.Phys-Manipulation, T.Abuse-Func	T.PHYSICAL	They match, as these threats are directed to the SCP.
T.Malfunction	T.FAULT	They match, as both dealing with parameter outside the normal operating conditions.
T.Leak-Inherent, T.Leak-Forced,	T.LEAKAGE	They match, as both dealing with leakage threats.
T.RND	T.RND	They match.

7.5 Security Objectives Mapping of the Platform

The following TOE security objectives are directly addressed to the platform: O.IC_SUPPORT, O.RECOVERY, O.OS.SUPPORT. These can be mapped to the Objective of the hardware platform as listed in the following table:

Platform Objectives TOE objectives	O.Phys-Manipulation	O.Phys-Probing	O.Malfunction	O.Leak-Inherent	O.Leak-Forced	O.Abuse-Func	O.Identification	O.RND	O.Add-Functions	O.Mem-Access	O.TDES	O.AES	O.SHA	O.Cap_Avail_Loader
O.IC_SUPPORT	X	X	X	X	X	X				X				X
O.RECOVERY			X											
O.OS_SUPPORT	X	X	X	X	X	X		X	X	X	X	X		X

O.IC_SUPPORT is mapped to the objectives of the platform features against physical attacks like physical probing and sophisticated analysis of the chip. This is all, but .O.Identification, O.Add-Functions, O.RND, O.TDES, O.AES and O.SHA.

O.RECOVERY is mapped to O.Malfunction as it is needed to restart the TOE after a malfunction.

O.OS_SUPPORT is mapped to all hardware relevant platform features that provide support for the functions to be not bypassed or altered. These are all, except .O.Identification.

The Platform Objective O.SHA is not relevant for this TOE and is therefore not mapped to any TOE objectives.

7.6 Separation of the Platform-TSF by TOE SFR

The TOE implements the following SFR, which are related to the platform SFR [ST_IC]. The following table lists the relevant Platform-SFRs being used by the Composite-ST. All not listed SFRs are irrelevant and not being used by the Composite-ST.

Hardware Platform SFR	TOE SFR	Remark
FPT_FLS.1	FPT_FLS.1	The SFR matches the platform SFR
FRU_FLT.2	FPT_RCV.3	The SFR matches the platform SFR
FPT_PHP.3	FPT_EMSEC.1	FPT_EMSEC.1 uses the platform SFR
FCS_RNG.1	FCS_RNG.1	The random number generator uses the platform TRNG as input
FCS_COP.1.1/TDES	FCS_COP.1.1/ACC_CYPHER FCS_COP.1.1/ACC_MAC	direct use of SCP
FCS_COP.1.1/AES	FCS_COP.1.1/ACC_CYPHER FCS_COP.1.1/ACC_MAC	direct use of SCP
FCS_COP.1/TDES_SCL	FCS_COP.1/DES FCS_COP.1.1/SCP FCS_COP.1.1/SCP-AUTH FCS_COP.1.1/SCP-KA FCS_COP.1.1/DES-MAC FCS_COP.1.1/RECEIPT	Library of hardware used
FCS_CKM.4/TDES_SCL FCS_CKM.4/AES_SCL	FCS_CKM.4	Library of hardware used
FCS_COP.1/AES_SCL	FCS_COP.1/AES FCS_COP.1.1/SCP FCS_COP.1.1/SCP-AUTH FCS_COP.1.1/SCP-KA FCS_COP.1.1/AES-MAC	Library of hardware used
FCS_COP.1/RSA	FCS_COP.1/RSA FCS_COP.1.1/DAP/TOKEN FCS_COP.1.1/RECEIPT	Library of hardware used

Hardware Platform SFR	TOE SFR	Remark
FCS_COP.1/ECDH	FCS_COP.1/ECDH FCS_COP.1.1/PACE_SUPP	Library of hardware used
FCS_COP.1/ECDSA	FCS_COP.1/ECDSA	EC library of the hardware is used
FCS_CKM.1/RSA-, FCS_CKM.1/EC	FCS_CKM.1	Key generation for RSA and ECC is supported by the platform
FDP_ACC.1	FDP_ACC.2/ADEL, FDP_ACC.2/FIREWALL FMT_MSA.1/EXT_MEM	Access control is supported by platform
FDP_ACF.1	FDP_ACF.1/ADEL, FDP_ACF.1/FIREWALL FMT_MSA.1/EXT_MEM	Security attribute based access control supported by platform
FMT_MSA.1	FMT_MSA.1/ADEL, FMT_MSA.1/EXT_MEM, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_MSA.1/CM, FMT_MSA.1/GPG	Management of security attributes supported by platform
FMT_MSA.3	FMT_MSA.3/ADEL, FMT_MSA.1/EXT_MEM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.3/CM, FMT_MSA.1/GPG	Static attribute initialization supported by platform

Hardware Platform SFR	TOE SFR	Remark
FDP_SDI.1, FDP_SDI.2	FDP_SDI.2	Stored data integrity monitoring and action supported by platform
FDP_ITT.1, FPT_ITT.1	FPT_EMSEC	TOE Emanation prevention supported by platform
FDP_IFC.1	FPT_EMSEC	TOE Emanation prevention supported by platform

ANNEX

References

[AIS20]	Anwendungshinweise und Interpretationen zum Schema, AIS 20: Funktionalitaetsklassen und Evaluationsmethodologie für deterministische Zufallszahlengeneratoren, Version 2.1, 02.12.2011, Bundesamt fuer Sicherheit in der Informationstechnik
[ANSI X9.62-2005]	Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 2005
[ANSI X9.63-2001]	Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport using Elliptic Curve Cryptography, November 20, 2001.
[APPNOTE]	APPLICATION NOTE CERTIFICATION OF “OPEN” SMART CARD

	PRODUCTS, Secrétariat général de la défense et de la sécurité nationale.
[BSI TR 03111]	BSI TR-03111 Elliptic Curve Cryptography, Version 2.0 2012-06-28
[CC1]	Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1. Revision 4. September 2012. CCMB-2012-09-001.
[CC2]	Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1. Revision 4. September 2012. CCMB-2012-09-002.
[CC3]	Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements. Version 3.1. Revision 4. September 2012. CCMB-2012-09-003
[CEM]	Common Methodology for Information Technology Security Evaluation, Evaluation Methodology. Version 3.1. Revision 4. September 2012. CEM-2012-09-004
[Conf_Guide]	Java Card Platform on SLE 78 (SLJ 52), Configuration Guide, E39028-02, Version 2.0 by Oracle Corporation (confidential).
[Conf_Tool]	Java Card Platform Implementation for Infineon on SLE 78 (SLJ 52GxxyyyzC), Configurator Tool Guide, E55697-02, Version 2.0 by Oracle Corporation (confidential)
[CSRS]	GlobalPlatform Card Security Requirements Specification, Version 1.0, May 2003.
[Data_Book]	Java Card Platform on SLE 78 (SLJ 52), Data Book, E39029-02, Version 2.0 by Oracle Corporation (confidential).
[ETSI SR 002-176]	Electronic Signatures and Infrastructures (ESI); Algorithms and Parameters for Secure Electronic Signatures. ETSI SR 002 176 V1.1.1:2003 - ETSI/TC ESI
[FIPS 180-4]	FIPS PUB 180-4 - FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION: Secure Hash Standard (SHS) , National Institute of Standards and Technology, March 2012
[FIPS 186-4]	FIPS PUB 186-4 - Digital Signature Standard (DSS), National Institute of Standards and Technology, Issued July, 2013
[FIPS 197]	FIPS PUB 197 - ADVANCED ENCRYPTION STANDARD (AES), National Institute of Standards and Technology, 2001-11-26
[FIPS140]	FIPS 140-2 - Security Requirements for Cryptographic Modules, National Institute of Standards and Technology, Issued May 25, 2001
[GP]	GlobalPlatform Card Specification, Version 2.2, March 2006.
[GP_A]	Confidential Card Content management – GlobalPlatform Card Specification v2.2 – Amendment A v1.0.1, January 2011
[GP_AM_D]	Secure Channel Protocol 03 – GlobalPlatform Card Specification v2.2 – Amendment D v1.1, September 2009
[GP_ID]	GlobalPlatform Card ID Configuration Version 1.0, December 2011 (GPC_GUI_039)
[ICAO Doc 9303]	"Doc 9303, Machine Readable Travel Documents, ICAO. Seventh edition 2015.

	Specifically Part 11: Security Mechanisms for MRTDs
[IEEE Std. 1363:2000]	IEEE Standard Specification for Public key Cryptography, IEEE Standards Board. The standard covers specification for public key cryptography including mathematical primitives for secret value deviation, public key encryption and digital signatures and cryptographic schemes based on those primitives.
[ISO/IEC 11770-3:2009]	INTERNATIONAL STANDARD ISO/IEC 11770-3:2008, TECHNICAL CORRIGENDUM 1, Published 2009-09-15, Information technology — Security techniques —Key management — Part 3: Mechanisms using asymmetric techniques
[ISO/IEC 14888-3:2009]	INTERNATIONAL STANDARD ISO/IEC 14888-3:2006, TECHNICAL CORRIGENDUM 2, Published 2009-02-15, Information technology — Security techniques — Digital signatures with appendix — Part 3: Discrete logarithm based mechanisms
[ISO/IEC 15946-2:2002]	Information technology -- Security techniques -- Cryptographic techniques based on elliptic curves -- Part 2: Digital Signatures (Withdrawn)
[ISO9797-1]	ISO/IEC 9797-1:2011 Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher : MAC algorithm 3.
[JAVASPEC]	The Java Language Specification. Third Edition, May 2005. Gosling, Joy, Steele and Bracha. ISBN 0-321-24678-0
[JCAPI3]	Java Card Platform, version 3.0.1 (May 2009), Classic Edition, Application Programming Interface, May 2009. Published by Oracle Corporation.
[JCBV]	Java Card Platform, version 2.2 Off-Card Verifier. June 2002. White paper. Published by Oracle Corporation.
[JCRE3]	Java Card Platform, version 3.0.1 (May 2009), Classic Edition, Runtime Environment (Java Card RE) Specification. May 2009. Published by Oracle Corporation.
[JCVM3]	Java Card Platform, version 3.0.1 (May 2009), Classic Edition, Virtual Machine (Java Card VM) Specification. Published by Oracle Corporation.
[JVM]	The Java Virtual Machine Specification. Lindholm, Yellin. ISBN 0-201-43294-3
[N890]	NIST Special Publication 800-90A: Recommendation for Random Number Generation Using Deterministic Random Bit Generators. January 2012, National Institute of Standards and Technology (NIST).
[PKCS V2.1]	Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1
[PP JCS]	Java Card Protection Profile Collection, Version 1.0b, August 2003, registered and certified by the French certification body (ANSSI) under the following references: [PP/0303] “Minimal Configuration”, [PP/0304] “Standard 2.1.1 Configuration”, [PP/0305] “Standard 2.2 Configuration” and [PP/0306] “Defensive Configuration”.
[PP]	Java Card Protection Profile - Open Configuration, Version 3.0, May 2012, Oracle Corporation.
[PP0017]	Machine Readable Travel Document with “ICAO Application", Basic Access Control, BSI-PP-0017-2005, BSI, Version 1.0, 2005-08-18

[PP0084b]	BSI-CC-PP-0084-2014, Security IC Platform Protection Profile with Augmentation Packages, Version 1.0, 13 January 2014
[SP800-108]	NIST Special Publication 800-108, Recommendation for Key Derivation Using Pseudorandom Functions, National Institute of Standards and Technology, October 2009
[SP800-38A]	NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation Methods and Techniques, National Institute of Standards and Technology, December 2001
[SP800-38B]	NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, National Institute of Standards and Technology, May 2005
[SP800-67]	NIST Special Publication 800-67, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, National Institute of Standards and Technology, Version 1.2, Revised July 2011
[ST_GP]	GlobalPlatform Smart Card Security Target Guidelines, V1.0, October 2005, GlobalPlatform Inc.
[ST_IC]	Public Security Target Lite M7892 Design Steps D11 and G12, Revision 1.7, 2016-11-16
[Tools_Prog]	Java Card Platform Implementation for Infineon on SLE 78 (SLJ 52GxxyyyzC), Tools Programming Guide, February 2016, E29342-02, Version 2.0. by Oracle Corporation
[TR03110]	Technical Guideline TR-03110, Advanced Security Mechanisms for Machine Readable Travel Documents, BSI, Version 2.10, 20. March 2012

ANNEX

Terms

AID	Application identifier, an ISO-7816 data format used for unique identification of Java Card applets (and certain kinds of files in card file systems). The Java Card platform uses the AID data format to identify applets and packages. AIDs are administered by the International Opens Organization (ISO), so they can be used as unique identifiers. AIDs are also used in the security policies (see “Context” below): applets’ AIDs are related to the selection mechanisms, packages’ AIDs are used in the enforcement of the firewall. Note: although they serve different purposes, they share the same namespace.
APDU	Application Protocol Data Unit, an ISO 7816-4 defined communication format between the card and the off-card applications. Cards receive requests for service from the CAD in the form of APDUs. These are encapsulated in Java Card System by the javacard.framework.APDU class ([JCAPI3]). APDUs manage both the selection-cycle of the applets (through Java Card RE mediation) and the communication with the Currently selected applet.
APDU buffer	The APDU buffer is the buffer where the messages sent (received) by the card depart from (arrive to). The Java Card RE owns an APDU object (which is a Java Card RE Entry Point and an instance of the javacard.framework.APDU class) that encapsulates APDU messages in an internal byte array, called the APDU buffer. This object is made accessible to the currently selected applet when needed, but any permanent access (out-of selection-scope) is strictly prohibited for security reasons.
Applet	The name given to any Java Card technology-based application. An applet is the basic piece of code that can be selected for execution from outside the card. Each applet on the card is uniquely identified by its AID.
Applet deletion manager	The on-card component that embodies the mechanisms necessary to delete an applet or library and its associated data on smart cards using Java Card technology.
BCV	The bytecode verifier is the software component performing a static analysis of the code to be loaded on the card. It checks several kinds of properties, like the correct format of CAP files and the enforcement of the typing rules associated to bytecodes. If the component is placed outside the card, in a secure environment, then it is called an off-card verifier. If the component is part of the embedded software of the card it is called an on-card verifier.
CAD	Card Acceptance Device or card reader. The device where the card is inserted, and which is used to communicate with the card. Unless explicitly said otherwise, in this document, CAD covers PCD.
CAP file	A file in the Converted applet format. A CAP file contains a binary representation of a package of classes that can be installed on a device and used to execute the package’s classes on a Java Card virtual machine. A CAP file can contain a user library, or the code of one or more applets.
Class	In object-oriented programming languages, a class is a prototype for an object. A class may also be considered as a set of objects that share a common structure and behavior. Each class declares a

	collection of fields and methods associated to its instances. The contents of the fields determine the internal state of a class instance, and the methods the operations that can be applied to it. Classes are ordered within a class hierarchy. A class declared as a specialization (a subclass) of another class (its super class) inherits all the fields and methods of the latter. Java platform classes should not be confused with the classes of the functional requirements (FIA) defined in the CC.
Context	A context is an object-space partition associated to a package. Applets within the same Java technology-based package belong to the same context. The firewall is the boundary between contexts (see "Current context").
Current context	The Java Card RE keeps track of the current Java Card System context (also called "the active context"). When a virtual method is invoked on an object, and a context switch is required and permitted, the current context is changed to correspond to the context of the applet that owns the object. When that method returns, the previous context is restored. Invocations of static methods have no effect on the current context. The current context and sharing status of an object together determine if access to an object is permissible.
Currently selected applet	The applet has been selected for execution in the current session. The Java Card RE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command from the CAD or PCD with this applet's AID, the Java Card RE makes this applet the currently selected applet over the I/O interface that received the command. The Java Card RE sends all further APDU commands received over each interface to the currently selected applet on this interface ([JCRE22], Glossary).
Default applet	The applet that is selected after a card reset or upon completion of the PICC activation sequence on the contactless interface ([JCRE22], §4.1)
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	Pre-issuance loaded software.
Firewall	The mechanism in the Java Card technology for ensuring applet isolation and object sharing. The firewall prevents an applet in one context from unauthorized access to objects owned by the Java Card RE or by an applet in another context.
Installer	The installer is the on-card application responsible for the installation of applets on the card. It may perform (or delegate) mandatory security checks according to the card issuer policy (for bytecode-verification, for instance), loads and link packages (CAP file(s)) on the card to a suitable form for the Java Card VM to execute the code they contain. It is a subsystem of what is usually called "card manager"; as such, it can be seen as the portion of the card manager that belongs to the TOE. The installer has an AID that uniquely identifies him, and may be implemented as a Java Card applet. However, it is granted specific privileges on an implementation-specific manner ([JCRE22],§10).
Interface	A special kind of Java programming language class, which declares methods, but provides no implementation for them. A class may be declared as being the implementation of an interface, and in this case must contain an implementation for each of the methods declared by the interface (See also shareable interface).
Java Card RE	The runtime environment under which Java programs in a smart card are executed. It is in charge of all the management features such as applet lifetime, applet isolation, object sharing, applet loading, applet initializing, transient objects, the transaction mechanism and so on.
Java Card RE Entry Point	An object owned by the Java Card RE context but accessible by any application. These methods are the gateways through which applets request privileged Java Card RE services: the instance methods associated to those objects may be invoked from any context, and when that occurs, a context switch

	to the Java Card RE context is performed. There are two categories of Java Card RE Entry Point Objects: Temporary ones and Permanent ones. As part of the firewall functionality, the Java Card RE detects and restricts attempts to store references to these objects.
Java Card RMI	Java Card Remote Method Invocation is the Java Card System version 2.2 and 3 Classic Edition mechanism enabling a client application running on the CAD platform to invoke a method on a remote object on the card. Notice that in Java Card System, version 3.0, the only method that may be invoked from the CAD is the process method of the applet class.
Java Card System	Java Card System includes the Java Card RE, the Java Card VM, the Java Card API and the installer.
Java Card VM	The embedded interpreter of bytecodes. The Java Card VM is the component that enforces separation between applications (firewall) and enables secure data sharing.
Logical channel	A logical link to an application on the card. A new feature of the Java Card System, version 2.2 and 3 Classic Edition, that enables the opening of simultaneous sessions with the card, one per logical channel. Commands issued to a specific logical channel are forwarded to the active applet on that logical channel. Java Card platform, version 2.2.2 and 3 Classic Edition, enables opening up to twenty logical channels over each I/O interface (contacted or contactless).
NVRAM	Non-Volatile Random Access Memory, a type of memory that retains its contents when power is turned off.
Object deletion	The Java Card System version 2.2 and 3 Classic Edition mechanism ensures that any unreferenced persistent (transient) object owned by the current context is deleted. The associated memory space is recovered for reuse prior to the next card reset.
PACE	Password Authenticated Connection Establishment.
Package	A package is a namespace within the Java programming language that may contain classes and interfaces. A package defines either a user library, or one or more applet definitions. A package is divided in two sets of files: export files (which exclusively contain the public interface information for an entire package of classes, for external linking purposes; export files are not used directly in a Java Card virtual machine) and CAP files.
PCD	Proximity Coupling Device. The PCD is a contactless card reader device.
PICC	Proximity Card. The PICC is a card with contactless capabilities.
RAM	Random Access Memory, is a type of computer memory that can be accessed randomly
SCP	Smart Card Platform. It is comprised of the integrated circuit, the native operating system and the dedicated software of the smart card.
Shareable interface	An interface declaring a collection of methods that an applet accepts to share with other applets. These interface methods can be invoked from an applet in a context different from the context of the object implementing the methods, thus “traversing” the firewall.
SIO	An object of a class implementing a shareable interface.
Subject	An active entity within the TOE that causes information to flow among objects or change the system’s status. It usually acts on the behalf of a user. Objects can be active and thus are also subjects of the TOE.
SWP	The Single Wire Protocol is a specification for a single-wire connection between the SIM card and a Near Field Communication (NFC) chip in a mobile handset

Transient object	An object whose contents are not preserved across CAD sessions. The contents of these objects are cleared at the end of the current CAD session or when a card reset is performed. Writes to the fields of a transient object are not affected by transactions.

ANNEX

Crypto Disclaimer

No.	Crypto-graphic Mechanism	Standard of Implementation	Key Size in Bits	Standard of Application	Comments	ST references (chapter)
Purpose: Authenticity						
1.	RSA[14] signature verification [15]	[PKCS V2.1] chapter 8.2 (RSASSA-PKCS1-v1_5) [FIPS 180-4] (SHA-1) [PKCS V2.1] chapter 8.1(RSASSA-PSS) [FIPS 180-4] (SHA-256)	RSASSA-PKCS1-v1_5: |Modulus| = (1024-2016 in increments of 32 bits), |Hash| = 160 RSASSA-PSS: |Modulus|=2048 bits, |Hash| = 256	[GP] App. C.2, C.3, C.6	FCO_NRO.2.3/CM (DAP- and Token-Verification	6.1.1.2 FCS_COP.1.1/DAP/TOKEN[16]
2.	Receipt signature generation[17] using RSA[18] and SHA-1	[PKCS V2.1] (RSA) Section 8.2 [FIPS 180-4] (SHA-1)	|Modulus| = 1024	[GP] App. C.5	generation of the Delegated Management Receipt signature attached to the	6.1.1.2 FCS_COP.1.1/RECEIPT[19]

14 RSASSA-PSS with SHA-256 for cryptographic key size of 2048 bits, RSASSA-PKCS1-v1_5 with SHA-1 for other cryptographic key sizes.

15 *verification of the DAP signature attached to Executable Load Files, verification of the "Delegated Management Token Signature"*

16 Used for a Delegated Management Token Signature, which *is a cryptographic value provided by a Card Issuer as proof that a Delegated Management operation has been authorized.*

17 *generation of the "Delegated Management Receipt signature" attached to the card management commands responses*

18 RSASSA-PKCS1-v1_5

19 Used for the Delegated Management Receipt Signature, which *is a cryptographic value provided by the card (if so required by the Card Issuer) as proof that a Delegated Management operation has occurred.*

No.	Crypto-graphic Mechanism	Standard of Implementation	Key Size in Bits	Standard of Application	Comments	ST references (chapter)
3.	Receipt signature generation using Retail MAC	[SP800-67] (TDES) [ISO9797-1] Section 7.3 (Retail MAC with DES)	| k | = 112	[GP] App. C.5	card management commands responses	
Purpose: Authentication						
4.	TDES in CBC mode[20]	[SP800-67] (TDES) [SP800-38A] (CBC) SCP02	|k| = 112 |host challenge| = 64[21] |card challenge| = 48[21]	[GP] App. E.4.2	SCP02	6.1.1.2 FCS_COP .1.1/SCP-AUTH
5.	KDF in counter mode with C-MAC as PRF[22]	[SP800-108] (KDF) [SP800-38B] (C-MAC) SCP03	|k| = 128 |host challenge| = 64[23] |card challenge| = 64[23]	[GP_AM_D] sec. 6.2.2.2, 6.2.2.3, 6.2.1, 4.1.5	SCP03	6.1.1.2 FCS_COP .1.1/SCP-AUTH
Purpose: Key Agreement[24]						
6.	TDES in CBC mode with ICV=0	[SP800-67] (TDES) [SP800-38A] (CBC) SCP02	|k|=112	[GP] App. E.4.1	SCP02	6.1.1.2 FCS_COP .1.1/SCP-KA
7.	KDF in counter mode with C-MAC as PRF	[SP800-108] (KDF) [SP800-38B] (C-MAC) SCP03	|k|=128	[GP_AM_D] sec. 6.2.1	SCP03	6.1.1.2 FCS_COP .1.1/SCP-KA
Purpose: Confidentiality						
8.	TDES in CBC mode	[SP800-67] (TDES)	|k| = 112	[GP] App. E.3, E.4		6.1.1.2

[20] [GP] E.4.1: *The DES operation used to generate these keys is always triple DES in CBC mode.*
[21] See [GP] sec. E.4.2.1
[22] [GP_AM_D] sec. 4.1.5: *Data derivation shall use KDF in counter mode as specified in NIST SP 800-108* [..]. *The PRF used in the KDF shall be C-MAC as specified in NIST SP 800-38B* [..]*, used with full 16 byte output length.*
[23] See [GP_AM_D] sec. 6.2.1
[24] Only key derivation techniques are used.

No.	Crypto-graphic Mechanism	Standard of Implementation	Key Size in Bits	Standard of Application	Comments	ST references (chapter)
9.	TDES in ECB mode	[SP800-38A] (CBC / ECB)			GlobalPlatform Secure Channels	FCS_COP .1.1/SCP
10.	AES in CBC mode	[FIPS 197] (AES) [SP800-38A] (CBC)	|k| = 128	[GP_AM_D] sec. 4.1.2, 6.2.6, 6.2.7	GlobalPlatform Secure Channels	6.1.1.2 FCS_COP .1.1/SCP
11.	TDES in CBC mode	[SP800-67] (TDES) [SP800-38A] (CBC)	|k| = 112	[ICAO Doc 9303] Part 11 chapter 9.8	secure messaging – encryption and decryption	6.1.1.2 FCS_COP .1.1/ACC_ CYPHER
12.	AES in CBC mode	[FIPS 197] (AES) [SP800-38A] (CBC)	|k| = 128, 192, 256	[ICAO Doc 9303] Part 11 chapter 9.8	secure messaging – encryption and decryption	6.1.1.2 FCS_COP .1.1/ACC_ CYPHER
Purpose: Integrity						
13.	TDES in Retail-MAC mode	[SP800-67] (TDES) [ISO9797-1] (Retail-MAC)	|k| = 112	[ICAO Doc 9303] Part 11 chapter 9.8	secure messaging – message authentication code	6.1.1.2 FCS_COP .1.1/ACC_ MAC
14.	AES in C-MAC mode (truncated to 64 bits)	[FIPS 197] (AES) [SP800-38B] (C-MAC)	|k| = 128, 192, 256	[ICAO Doc 9303] Part 11 chapter 9.8	secure messaging – message authentication code	6.1.1.2 FCS_COP .1.1/ACC_ MAC
15.	TDES in Retail-MAC mode (SCP02)	[SP800-67] (TDES) [ISO9797-1] (Retail-MAC)	|k| = 112	[GP] App. E.3.2,E3.3	with options 0x15, 0x1a, or 0x55	6.1.5 FTP_ITC.1 /CM
16.	AES in C-MAC mode (truncated to 64 bits[25]) (SCP03)	[FIPS 197] (AES) [SP800-38B] (C-MAC)	|k| = 128, 192, 256	[GP_AM_D]	with options 0x00 or 0x10	6.1.5 FTP_ITC.1 /CM

[25] [GP_AM_D] sec. 6.2.4: *The Secure channel shall support a MAC of 8 bytes length (even if the AES block length is 16 bytes). Hence the 8 most significant bytes are considered.*

No.	Crypto-graphic Mechanism	Standard of Implementation	Key Size in Bits	Standard of Application	Comments	ST references (chapter)
Purpose: Trusted Channel						
17.	SCP02	[GP] App. E[26]	-	[GP] App. E	with options 0x15, 0x1a, or 0x55	6.1.5 FTP_ITC.1/CM
18.	SCP03	[GP_AM_D][27]	-	[GP_AM_D]	with options 0x00 or 0x10	6.1.5 FTP_ITC.1/CM

No.	Crypto-graphic Mechanism	Standard of Implemen-tation	Key Size in Bits	Security Level above 100 Bits	Comments	ST references (chapter)
Purpose: Cryptographic Primitives						
TDES						
19.	TDES in ECB mode	[SP800-67] (TDES) [SP800-38A] (ECB / CBC)	|k| = 112, 168	no	-	6.1.1.2 FCS_COP.1.1/DES
20.	TDES in CBC mode		|k| = 112	no	-	
21.			|k| = 168	yes	-	
AES						
22.	AES in ECB mode	[FIPS 197] (AES) [SP800-38A] (ECB / CBC)	|k| = 128, 192, 256	no	-	6.1.1.2 FCS_COP.1.1/AES
23.	AES in CBC mode		|k| = 128, 192, 256	yes	-	

26 Additionally cf. lines 4, 6, 8, 9, 15,

27 Additionally cf. lines 5, 7, 10, 16

No.	Crypto-graphic Mechanism	Standard of Implemen-tation	Key Size in Bits	Security Level above 100 Bits	Comments	ST references (chapter)
RSA						
24.	RSA data encryption	[PKCS v2.1]	|Modulus| > 1976 (1984, 2016, 2048 bits)	yes	RSAEP section 5.1.1 in [PKCS v2.1], without 5.1.1.1	6.1.1.2 FCS_COP.1.1/RSA 6.1.1.2 FCS_CKM.1.1/RSA
25.	RSA Key generation	[PKCS v2.1]	|Modulus| > 1976 (1984, 2016, 2048 bits)	yes	According to section 3.2(2) in [PKCS V2.1], for u=2, i.e., without any (r_i, d_i, t_i), i > 2. For p x q < $2^{2048+64}$ additionally according to section 3.2(1)]	
26.	RSA data decryption with or without RSA-CRT	[PKCS v2.1]	|Modulus| > 1976 (1984, 2016, 2048 bits)	yes	RSADP (section 5.1.2 in [PKCS v2.1] for u = 2, only supported up to n < $2^{2048+64}$)	
27.	RSA signature generation with or without RSA-CRT with encoding[28] using SHA[29]	[PKCS V2.1] (RSA) [FIPS 180-4] (SHA)	|Modulus| > 1976 (1984, 2016, 2048 bits)	yes	RSASSA-PKCS-v1_5 (section 8.2.1 in [PKCS v2.1]);	
28.	RSA signature verification with encoding[30] using SHA[31]	[PKCS V2.1] (RSA) [FIPS 180-4] (SHA)	|Modulus| > 1976 (1984, 2016, 2048 bits)	yes	RSASSA-PKCS-v1_5 (section 8.2.2 in [PKCS v2.1]).	

28 RSASSA-PKCS-v1_5
29 Cf. line 38
30 RSASSA-PKCS-v1_5
31 Cf. line 38

29.	RSA signature generation with or without RSA-CRT without encoding[32]	[PKCS V2.1] (RSA)	|Modulus| > 1976 (1984, 2016, 2048 bits)	yes	RSASP1 (section 5.2.1 in [PKCS v2.1] for u = 2, only supported up to n < $2^{2048+64}$)	
30.	RSA signature verification without encoding[33]	[PKCS V2.1] (RSA)	|Modulus| > 1976 (1984, 2016, 2048 bits)	yes	RSAVP1 (section 5.2.2 in [PKCS v2.1] - without 5.2.2.1	
ECDSA						
31.	ECDSA signature generation and verification (ECDSA-FP)	[ANSI X9.62 – 2005] / [ISO/IEC 14888-3:2009] / [IEEE Std. 1363:2000]	|k| = 192[34]	no	Signature Generation[35] Signature Verification[36]	6.1.1.2 FCS_COP.1.1/ECDSA
32.			|k| = 224, 256, 320, 384, 512, 521[37]	yes		
33.	ECDSA signature generation and verification (ECDSA-F2M)		|k| = 233, 283, 409[38]	yes		
34.			|k| = 192[34]	no		6.1.1.2

32 RSASP1

33 RSAVP1

34 [FIPS 186-4]: P-{192} and [Brainpool RFC 5639]: brainpoolP{192}r1 and brainpoolP{192}t1

35 1.According to section 7.3 in [ANSI X9.62 – 2005]. Not implemented is step d) and e) thereof. The output of step e) has to be provided as input to our function by the caller. Deviation of step c) and f): The jumps to step a) were substituted by a return of the function with an error code, the jumps are emulated by another call to our function. 2. According to sections 6.4.3 Signature Process in [ISO/IEC 14888-3:2009] Chapter 6.4.3.3 is not supported Chapter 6.4.3.5 is not supported - the hash-code of H of the message has to be provided by the caller as input for our function. Chapter 6.4.3.7 is not supported Chapter 6.4.3.8 is not supported; 3. According to section 7.2.7 ECSP-DSA in [IEEE Std. 1363:2000] Deviation of step (3) and (4): - The jumps to step 1 were substituted by a return of the function with an error code, the jumps are emulated by another call to our function.

36 1.According to section 7.4.1 in ANSI X9.62–2005. Not implemented is step b) and c) thereof. The output of step c) has to be provided as input to our function by the caller. Deviation of step d): Beside noted calculation, our algorithm adds a random multiple of BasepointerOrder n to the calculated values u1 and u2. 2. According to sections 6.4.4 Signature Verification Process in [ISO/IEC 14888-3:2009] Chapter 6.4.4.2 is not supported, Chapter 6.4.4.3 is not supported: - The hash-code H of the message has to be provided by the caller as input to our function; 3. According to section 7.2.8 ECVP-DSA in [IEEE Std. 1363-:2000].

37 [FIPS 186-4]: P-{224, 256, 384, 521} and [Brainpool RFC 5639]: brainpoolP{224, 256, 320, 384,512}r1 and brainpoolP{224, 256, 320, 384,512}t1

38 [FIPS 186-4]: Curve K-{233,283,409} and Curve B-{233,283, 409}

	EC key generation	[ANSI X9.62 – 2005] / [ISO/IEC 14888-3:2009] / [IEEE Std. 1363:2000]	|k| = 224, 256, 320, 384, 512, 521[37]	yes	([ANSI X9.62 – 2005] appendix A4.3) ([ISO/IEC 14888-3:2009] section 6.4.2) ([IEEE Std. 1363:2000] appendix A.16.9)	FCS_CKM.1.1/EC
			|k| = 233, 283, 409[38]	yes		
ECDH						
35.	ECDH	[ANSI X9.63-2001] [ISO/IEC 11770-3:2009] [IEEE Std. 1363:2000]	|k| = 192 [34]	no	According to section 5.4.1 in [ANSI X9.63 -2001] Unlike section 5.4.1.3 the implementation does not only return the x-coordinate of the shared secret, but rather the x-coordinate and y-coordinate According to section Appendix D.6 Key agreement of Diffie-Hellmen type in [ISO/IEC 11770-3:2009] the function enables the operations described in appendix D.6. According to section 7.2.1 ECSVHDP-DP in [IEEE Std. 1363:2000] Unlike section 7.2.1 our implementation not only returns the x-coordinate of the shared secret, but rather the x-coordinate and the y-coordinate.	6.1.1.2 FCS_COP.1.1/ECDH
36.			|k| = 224, 256, 320, 384, 512, 521[37] and 233, 283, 409[38]	yes		
SHA						
37.	SHA-1	[FIPS 180-4]	|Hash| = 160	No	No check of maximum input length according FIPS 180-4 is implemented.	6.1.1.2 FCS_COP.1.1/SHA
38.	SHA-2	[FIPS 180-4]	|Hash| = 224, 256, 384, 512	yes	No check of maximum input length according FIPS 180-4 is implemented.	6.1.1.2 FCS_COP.1.1/SHA
RNG						

39.	RNG (DRG.4)	[AIS20], [N890 Ch 10.2.1]	|k| = 128	yes	Chapter 11.3 of [N890] is not implemented	6.1.8 FCS_RNG.1
ECC Basic Operations						
40.	Point Addition	[BSI TR 03111]	|k| = 192,	No	-	6.1.1.2 FCS_COP.1.1/PACE_SUPP
			|k| = 224, 233, 256, 283, 320, 384, 409, 512, 521	Yes		
41.	Scalar Multiplication		|k| = 192	No	Scalar multiplication according to section 4.3.1 in [BSI TR 03111] without optional Z_AB	
			|k| = 224, 233, 256, 283, 320, 384, 409, 512, 521	Yes		
MACs						
42.	Retail MAC	[SP800-67] [ISO/IEC9797-1]	|k| = 112	No	MAC algorithm 3 / padding method 2	6.1.1.2 FCS_COP.1.1/DES-MAC
43.	AES CMAC	[FIPS 197] [SP 800-38B]	|k| = 128, 192, 256	Yes	-	6.1.1.2 FCS_COP.1.1/AES-MAC