NXP J3E081_M64_DF, J3E081_M66_DF, J3E041_M66_DF, J3E016_M66_DF, J3E041_M64_DF and J3E016_M64_DF Secure Smart Card Controller Revision 3

Security Target Lite

Rev. 00.03 — 13 August 2013
NSCIB-CC-13-37762

Evaluation documentation
PUBLIC

Document information

Info	Content
Keywords	JCOP, ST, Security Target Lite
Abstract	This is the Security Target Lite for JCOP v2.4.2 Revision 3. It defines the Contract for the certification according to Common Criteria.

Revision history

Rev	Date	Description
00.01	20130725	Derived from Security Target
00.02	20130803	Fixed typo in FCS_RNG
00.03	20130813	Adapted section 6.2 to reflect EAL 4 requirements

Glossary

A.xxx	Assumptions
AID	Application identifier, an ISO-7816 data format used for unique identification of Java Card applications (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 Standards 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 name space.
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 ([19]). APDUs manage both the selection-cycle of the applets (through JCRE 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 JCRE owns an APDU object (which is a JCRE 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 is given to a Java Card technology-based user 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.

BSI	"Bundesamt für Sicherheit in der Informationstechnik", German national certification body
CAD	Card Acceptance Device, or card reader. The device where the card is inserted, and which is used to communicate with the card.
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.
CC	Common Criteria
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.
CM	Card Manger
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 JCRE 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 JCRE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command from the CAD with this applet's AID, the JCRE makes this applet the currently selected applet. The JCRE sends all APDU commands to the currently selected applet ([20] Glossary).
Default applet	The applet that is selected after a card reset ([20], §4.1).
DCSSI	"*Direction Centrale de la Sécurité des Systèmes d'Information*", French national certification body

EAL	Evaluation Assurance Level
EEPROM	Electrically Erasable Programmable ROM
Embedded Software	Pre-issuance loaded software.
ES	Embedded 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 JCRE or by an applet in another context.
HAL	Hardware Abstraction Layer
IC	Integrated Circuit
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 JCVM 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 ([20], §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).
JCRE	The Java Card runtime environment consists of the Java Card virtual machine, the Java Card API, and its associated native methods. This notion concerns all those dynamic features that are specific to the execution of a Java program in a smart card, like applet lifetime, applet isolation and object sharing, transient objects, the transaction mechanism, and so on.
JCRE Entry Point	An object owned by the JCRE context but accessible by any application. These methods are the gateways through which applets request privileged JCRE system services: the instance methods associated to those objects may be invoked from any context, and when that occurs, a context switch to the JCRE context is performed. There are two categories of JCRE Entry Point Objects: Temporary ones and Permanent ones. As part of the firewall functionality, the JCRE detects and restricts attempts to store references to these objects.
JCRMI	Java Card Remote Method Invocation is the Java Card System, version 2.2.2, 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 2.1.1, the only method that may be invoked from the CAD is the `process` method of the `applet` class.
Java Card System	The Java Card System: the JCRE (JCVM +API), the installer, and the on-card BCV (if the configuration includes one).
JCVM	The embedded interpreter of bytecodes. The JCVM 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 2.2.2, that enables the opening of up to four 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.
MMU	Memory management unit
NOS	Native Operating System. For this ST, NOS means the TOE without the underlying hardware platform, i.e. NOS is equivalent to the smart card embedded software
OT.xxx	Security objectives for the TOE
Object deletion	The Java Card System, version 2.2.2, 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.
OE.xxx	Security objectives for the environment
OSP.xxx	Organizational security policies
Package	A package is a name space 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.
SCP	Smart card platform. It is comprised of the integrated circuit, the operating system and the dedicated software of the smart card.
PP	Protection Profile
RAM	Random Access Memory
RMI	Remote Method Invocation
ROM	Read Only Memory
RTE	Runtime Environment
SC	Smart Card
SF.xxx	Security function

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.
SOF	Strength Of Function
ST	Security Target
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.
T.xxx	Threats
TOE	Target of Evaluation
Transient object	An object whose contents is 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.
TSF	TOE Security Functions
User	Any application interpretable by the JCRE. That also covers the packages. The associated subject(s), if applicable, is (are) an object(s) belonging to the `javacard.framework.applet` class.
VM	Virtual Machine

1. ST Introduction (ASE_INT)

1.1 ST reference and TOE reference

Table 1. ST reference and TOE reference

Title	NXP J3E081_M64_DF, J3E081_M66_DF, J3E041_M66_DF, J3E016_M66_DF, J3E041_M64_DF and J3E016_M64_DF Secure Smart Card Controller Revision 3 Security Target Lite
Version	Rev. 00.03
Date	13 August 2013
Author(s)	NXP Semiconductors
Developer	NXP Semiconductors
Product Type	Java Card
TOE name/version	NXP J3E081_M64_DF, J3E081_M66_DF, J3E041_M66_DF, J3E016_M66_DF, J3E041_M64_DF and J3E016_M64_DF Secure Smart Card Controller Revision 3
Certification ID	NSCIB-CC-13-37762
TOE hardware	P5CD081V1D
CC used	Common Criteria for Information Technology Security Evaluation Version 3.1, Revision 4, September 2012 (Part 1, Part 2 and Part 3)

1.2 TOE overview

This document details the security target lite for NXP J3E081_M64_DF, J3E081_M66_DF, J3E041_M66_DF, J3E016_M66_DF, J3E041_M64_DF and J3E016_M64_DF Secure Smart Card Controller Revision 3 (also named JCOP 2.4.2 R3). It is compliant to the protection profile “Java Card System - Open Configuration Protection Profile, Version 2.6, Certified by ANSSI, the French Certification Body April, 19th 2010” [5].

The ST fulfils all requirements of [5]. This ST chooses a hierarchically higher EAL, namely EAL4, augmented by ALC_DVS.2, AVA_VAN.5, and ASE_TSS.2.

The basis of this composite evaluation is the composite evaluation of the hardware and the cryptographic library. Table 2 gives the details of the underlying evaluations of the cryptographic library and the underlying hardware platforms. For the hardware evaluation no maintenance report is applicable. The hardware is compliant to the protection profile “Smartcard IC Platform Protection Profile (SSVG-PP), Version 1.0, June 2007; registered and certified by (BSI) under the reference BSI-PP-0035-2007“ [6].

Table 2. Underlying evaluations

Cert ID	Name	Reference
BSI-DSZ-CC-0864	Crypto Library V2.7/V2.9 on SmartMX P5Cx081/ CD041/ CD021/ CD016 V1D, Security Target, Rev. 1.3, 08 May 2013, BSI-DSZ-CC-0864	**[9]**

Cert ID	Name	Reference
BSI-DSZ-CC-0707	NXP Secure Smart Card Controllers P5CD016V1D / P5CD021V1D / P5CD041V1D / P5Cx081V1D, Security Target Lite, NXP Semiconductors, Revision 1.1, BSI-DSZ-CC-0707, 24. October 2011	[10]

For the P5CD081V1D hardware of this TOE three minor configuration options can be freely chosen during Smartcard IC Personalization (see section 2.2.5 of the Hardware Security Target [10]):

- “MIFARE DESFire Emulation = A” in which the DESFire interface is disabled
- “MIFARE DESFire Emulation = D2” in which the DESFire interface is enabled and 2KB DESFire EEPROM memory is reserved
- “MIFARE DESFire Emulation = D4” in which the DESFire interface is enabled and 4KB DESFire EEPROM memory is reserved
- “MIFARE DESFire Emulation = D8” in which the DESFire interface is enabled and 8KB DESFire EEPROM memory is reserved

From [6] relevant requirements for the hardware platform were taken. The relevant requirements for the Java Card functionality were taken from [5]. In addition for this version the relevant requirements for the MIFARE DESFire emulation where taken from [10].

JCOP 2.4.2 R3 is based on Java Card 3.0.1 and Global Platform 2.2.1 industry standards, and allows post-issuance downloading of applications that have been previously verified by an off-card trusted IT component. It implements high security mechanisms and supports various protocols, cryptographic algorithms, and the Secure Box, see Section 1.3.1.

1.3 TOE description

This part of the document describes the TOE to provide an understanding of its security requirements, and addresses the product type and the general IT features of the TOE.

1.3.1 TOE abstract and definition

The target of evaluation (TOE) is the JCOP 2.4.2 R3. It consists of:

- Smart card platform (SCP)
  (parts of the hardware platform and hardware abstraction layer)
- Embedded software
  (Java Card Virtual Machine, Runtime Environment, Java Card API, Card Manager)
- Native MIFARE DESFire application
  (physically always present but logical availability depends on configuration (see section 2.2.5 of the HW Security Target [10]))

The TOE does not include any software on the application layer (Java Card applets). This is shown schematically in Fig 1.

The Smart Card Platform (SCP) consists of the Hardware Abstraction Layer (HAL) and the Hardware Platform. The cryptographic library (Crypto Library) is part of the Hardware Abstraction Layer (HAL). Not all functionality of the Crypto Library is used by the Embedded Software, this unused functionality is not linked with the code and is therefore not part of the HAL. All functions in the HAL are used by the TOE. Not all functionality of the Hardware Platform is used for the TOE functionality and exposed at external interfaces.

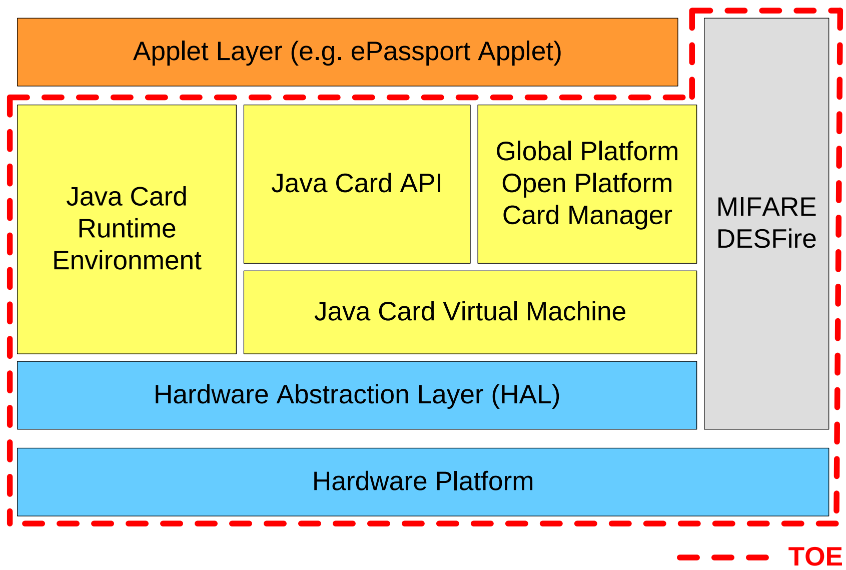


Fig 1. JCOP Architecture

The Java Card virtual machine (JCVM) is responsible for ensuring language-level security; the JCRE provides additional security features for Java Card technology-enabled devices.

The basic runtime security feature imposed by the JCRE enforces isolation of applets using an applet firewall. It prevents objects created by one applet from being used by another applet without explicit sharing. This prevents unauthorized access to the fields and methods of class instances, as well as the length and contents of arrays.

The applet firewall is considered as the most important security feature. It enables complete isolation between applets or controlled communication through additional mechanisms that allow them to share objects when needed. The JCRE allows such sharing using the concept of “shareable interface objects” (SIO) and static public variables. The JCVM should ensure that the only way for applets to access any resources are either through the JCRE 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 [21] on the byte codes and the correctness of the CAP file format are satisfied).

The Card Manager is conformant to the Global Platform Card Specification 2.2.1 [15] and is responsible for the management of applets in the card. For the present TOE the post issuance of applets is allowed. For more details of the Java card functionality see Section 1.3.5.

The native application MIFARE DESFire (grey box in Fig 1) is logically only available in the Minor Configuration options “MIFARE DESFire Emulation = D2”, “MIFARE DESFire Emulation = D4”, and “MIFARE DESFire Emulation = D8”. In the Minor Configuration option “MIFARE DESFire Emulation = A”, the grey box is not available in the hardware.

The Java card design and implementation is based on the Java Card 3.0.1 and on the GlobalPlatform 2.2.1 industry standards. The following features comprise the logical scope of the TOE:

- 3 different communication protocols:
    a. ISO 7816 T=1
    b. ISO 7816 T=0
    c. ISO 14443 T=CL (contact-less)
- Cryptographic algorithms and functionality:
    a. 3DES (112 and 168 bit keys) for en-/decryption (CBC and ECB) and MAC generation and verification (Retail-MAC, CMAC and CBC-MAC)
    b. AES (Advanced Encryption Standard) with key length of 128, 192, and 256 Bit for en-/decryption (CBC and ECB)) and MAC generation and verification (CMAC, CBC-MAC)
    c. RSA and RSA CRT (1976 up to 2048 bits keys) for en-/decryption and signature generation and verification
    d. RSA and RSA CRT key generation (1976 up to 2048 bits keys)
    e. SHA-1, SHA-224, and SHA-256 hash algorithm
    f. ECC over GF(p) algorithm that can be used for signature generation and signature verification (ECDSA) from 128 to 320 bits.
    g. ECC over GF(p) key generation algorithm that can be used to generate ECC over GF(p) key pairs.
    h. Random number generation according to class DRG.3 and DRG.2 of AIS 20 [8].
    i. Secure point addition for Elliptic Curves over GF(p).
    j. Diffie-Hellman key agreement and EC-DH of GF(p).
- Java Card 3.0.1 functionality:
    a. Garbage Collection fully implemented with complete memory reclamation incl. compactification
    b. Support for Extended Length APDUs
- GlobalPlatform 2.2.1 functionality:
    a. CVM Management (Global PIN) fully implemented: all described APDU and API interfaces for this feature are present
    b. Secure Channel Protocol (SCP01, SCP02, and SCP03 (only in Mask 64)) is supported
    c. Card manager
    d. Delegated management
- Proprietary SM Accelerator Interface, secure massaging API of JCOP 2.4.2 R3. The purpose of this API is to increase the performance of the secure messaging. It is specially designed for LDS applets which are used for the electronic passport as defined by ICAO.
- Post-issuance installation and deletion of applets, packages and objects
- Pre-personalization mechanism

- A Secure Box concept is implemented within JCOP 2.4.2 R3. The Secure Box is a construct which allows to run non certified third party native code and ensures that this code cannot harm, influence or manipulate the JCOP 2.4.2 R3 operating system or any of the applets executed by the operating system.The separation of the native code in the Secure Box from other code and/or data residing on the hardware is ensured by the Hardware MMU which has been certified in the hardware evaluation (see [10]).
- MIFARE DESFire application accessible via contactless interface and via Java Card API (availability depends on configuration and hardware)

1.3.2 Non-TOE hardware/software/firmware

- In order to communicate, the TOE has to be connected to a terminal that supports the ISO7816 or ISO14443 protocols. In order to communicate ISO14443 the TOE may be connected to an antenna or appropriate communication interface (e.g. S^2C) which is not part of the scope of this evaluation. It is noted that the TOE fulfils its security functions independent of the terminal or other communication interface.

1.3.3 TOE Life-Cycle

The life-cycle for this Java Card is based on the general smart card life-cycle defined in the Smart Card IC PP [6] and has been adapted to Java Card specialties. The main actors are marked with bold letters.

Table 3. TOE Life Cycle

Phase	Name	Description
1	IC Embedded Software Development	The IC Embedded Software Developer is in charge of • smartcard embedded software development including the development of Java applets and • specification of IC pre-personalization requirements, though the actual data for IC pre-personalization come from phase 4,5, or 6.
2	IC Development	The IC Developer • designs the IC, • develops IC Dedicated Software, • provides information, software or tools to the IC Embedded Software Developer, and • receives the smartcard embedded software from the developer, through trusted delivery and verification procedures. From the IC design, IC Dedicated Software and Smartcard Embedded Software, the IC Developer • constructs the smartcard IC database, necessary for the IC photomask fabrication.
3	IC Manufacturing	The **IC Manufacturer** is responsible for • producing the IC through three main steps: IC manufacturing, IC testing, and IC pre-

Phase	Name	Description
		personalization The IC Mask Manufacturer • generates the masks for the IC manufacturing based upon an output from the smartcard IC database
4	IC Packaging	The IC Packaging Manufacturer is responsible for • IC packaging and testing.
5	Composite Product Integration	The Composite Product Manufacturer is responsible for • smartcard product finishing process including applet loading and testing.
6	Personalization	The **Personalizer** is responsible for • smartcard (including applet) personalization and final tests. Applets may be loaded onto the chip at the personalization process.
7	Operational Usage	The Consumer of Composite Product is responsible for • smartcard product delivery to the smartcard end-user, and the end of life process. • applets may be loaded onto the chip

The evaluation process is limited to phases 1 to 6.

Applet development is outside the scope of this evaluation.

Applets can be loaded into ROM or EEPROM.

Applet loading into ROM can only be done in phase 3. Applet loading into EEPROM can be done in phases 3, 4, 5, and 6.

Applet loading in phase 7 is also allowed. This means post-issuance loading of applets can be done for a certified TOE.

It is possible to load patch code into EEPROM in phases 3, 4, 5, and 6. The certification is only valid for the ROM code version and the patch code version (if applicable) as stated in Table 4.

The delivery process from NXP to their customers (to phase 4 or phase 5 of the life cycle) guarantees, that the customer is aware of the exact versions of the different parts of the TOE as outlined above.

TOE documentation is delivered in electronic form (encrypted) according to defined mailing procedures.

Note: *The TOE development and manufacturing environment (phases 1 to 3) is in the scope of this ST. These phases are under the TOE developer scope of control. Therefore, the objectives for the environment related to phase 1 to 3 are covered by Assurance measures, which are materialized by documents, process and procedures evaluated through the TOE evaluation process.*
The `product usage phases` (phase 4 to 7) are not in the scope of the evaluation. During these phases, the TOE is no more under the developer control. In this environment, the TOE protects itself with its own Security functions. But some additional usage recommendation must also be

followed in order to ensure that the TOE is correctly and securely handled, and that shall be not damaged or compromised. This ST assumes (A.USE_DIAG, A.USE_KEYS) that users handle securely the TOE and related Objectives for the environment are defined (OE.USE_DIAG, OE.USE_KEYS).

1.3.4 TOE Identification

The delivery comprises the following items:

Table 4. Delivery Items

Type	Name	Version	Date
Hardware	NXP J3E081_M64_DF, J3E081_M66_DF, J3E041_M66_DF, J3E016_M66_DF, J3E041_M64_DF and J3E016_M64_DF Secure Smart Card Controller Revision 3 ROM Code (Mask ID) Patch Code (Patch ID)	see Table 5	
Document	User Manual (AGD_OPE) for the applet developer [33]	0.6	16th July 2013
Document	Administrator Manual (AGD_PRE) [34]	0.5	16th July 2013
Document	HW Data Sheet [17]	0.2	26th March 2013
Document	Secure Box User Manual [36]	3.4	18th March 2013
(optional) Document[1]	HW Guidance Manual [11]	3.0	10th January 2012

Table 5 lists the product identification for all products covered by this security target.

Table 5. Product Identification

Product	Mask ID	Mask Name	Patch ID
J3E081_M64_DF J3E041_M64_DF J3E016_M64_DF	64	NX250B	01
J3E081_M66_DF J3E041_M66_DF J3E016_M66_DF	66	NX250C	01

Note: Differences between Mask 64 and Mask 66:

The difference between Mask 64 and Mask 66 is that in Mask 66 the FIPS Selftest API is not implemented, and no SCP03 implementations are available. Both configurations support the same set of SFR’s.

[1] The hardware guidance manual is only required for developers of native libraries.

The commercial product names of JCOP products have the following form.

Jabcccxdd(d)/mvsrrff[o]

In case of a pure contact product (a=1 or a=2), the option field “o” is absent. Pure contact products cannot support MIFARE DESFire. With respect to MIFARE DESFire these products correspond to contactless products in Config A (o=0).

The ’**J**’ is constant, the other letters are variables. For a detailed description of these variables, please see Table 7.

For the certified products some variables need to have defined settings. These settings are given in Table 6

Table 6. Products commercial names

Variable	Must have one of these values (details see Table 7)
a	3
b	E
ccc	081, 041, 016 (016 and 041 represent the J3E016 and J3E041 platforms which are derivates of the J3E081 platform with limited EEPROM sizes).
x	Depends on the application of possible applets in ROM. A letter can be chosen (e.g. V for Visa).
dd	These 2 letters indicate the package. All package types which are covered by the certification of the used hardware are allowed. For the list of certified packages please refer to the public security target of the corresponding hardware [10].
M	T
Vs	1D:
O	F, B, C, D: for J3E081_M64_DF, J3E081_M66_DF, J3E041_M66_DF, J3E016_M66_DF, J3E041_M64_DF, J3E016_M64_DF for a=2: variable o is absent

The values for ’rr’, ’ff’ are customer dependent.

The following table explains the naming conventions of the commercial product name of the JCOP products. Every JCOP product gets assigned such a commercial name, which includes also customer and application specific data. This table does not give any information about which commercial products are Common Criteria certified.

Table 7. JCOP Commercial Name Format

Variable	Meaning	Example Values	Parameter settings
a	Hardware Type	1	SC hardware (no PKI, no contactless interface)
		2	CC hardware (no contactless interface)
		3	CD hardware

Variable	Meaning	Example Values	Parameter settings
		4	USB hardware
		5	NFC (S^2C) hardware
		6	CL hardware for µSD
		7	Authentication (I^2C and/or SPI)
b	JCOP version	A	JCOP V2.4.1 R3
		C	JCOP V2.4.2 R1
		D	JCOP V2.4.2 R2
		E	JCOP V2.4.2 R3
		G	JCOP V3.0
ccc	EEPROM size in KB	081	80[2] KB EEPROM
x	JCOP type	G	Generic
		C	Customized
		others	others are possible and are application dependent
dd(d)	Delivery type	U0	729µm unsawn unthinned wafer, inkless
		UA	150µm sawn wafer, inkless
		UE	75µm sawn wafer, inkless
		XS	PDM/PCM module
		XT	PDM/PCM – Pd (Silver)
		A4	MOB4 (not for P5CD081)
		A6	MOB6
		HN1	HVQFN32 package
		others	other delivery forms
m	Manufacturing Site Code	T, S	
v	Silicon Version Code	0, 1	
s	Silicon Version Subcode	B, A	
rr	ROM Code ID		
ff	FabKey ID		
o	Option	E	Config A (MIFARE Flex with No MIFARE Classic)
		3	Config B1 (MIFARE FleX with MIFARE Classic 1K)
		6	Config B4 (MIFARE FleX with

[2] With the introduction of the P5Cx081 family the EEPROM size of the product name has been increased by one to indicate the new family. This means that P5Cx081 only has 80 KB EEPROM and the P5Cx145 has only 144 KB EEPROM.

Variable	Meaning	Example Values	Parameter settings
			MIFARE Classic 4K)
		F	Config A (No MIFARE DESFire)
		B	Config D2 (MIFARE DESFire 2K)
		C	Config D4 (MIFARE DESFire 4K)
		D	Config D8 (MIFARE DESFire 8K)

1.3.5 Java Card Technology

For an overview on Java Card technology the reader is referred to Section 2 of the Java Card Protection Profile [5].

In the Java Card Protection Profile, the Java Card System is divided into so-called groups. For a detailed explanation of these groups please see the Java Card Protection Profile [5].

For the TOE of this certification the groups marked with ‘TOE’ are part of the TOE evaluation. Groups marked with ‘IT’ are considered in the TOE IT environment, and groups marked with ‘—‘ are out of scope of this evaluation.

Table 8. TOE Groups Overview

Group	Description	Scope
Core (CoreG)	The CoreG contains the basic requirements concerning the runtime environment of the Java Card System, such as the firewall policy and the requirements related to the Java Card API. This group is within the scope of evaluation.	TOE
Smart card platform (SCPG)	The SCPG contains the security requirements for the smart card platform, that is, operating system and chip that the Java Card System is implemented upon. In the present case, this group applies to the TOE and is within the scope of evaluation.	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 features, which provides a new protocol of communication between the terminal and the applets. This group is not implemented and therefore outside the scope of evaluation.	-
Logical channels (LCG)	The LCG contains the security requirements for the logical channels, which provide a runtime environment where several applets can be	-

Group	Description	Scope
	simultaneously selected or a single one can be selected more than once. This group is not within the scope of evaluation.	
Object deletion (ODELG)	The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism.	TOE
Bytecode verification (BCVG)	The BCVG contains the security requirements concerning the bytecode verification of the application code to be loaded on the card. In the present case, this group of SFRs applies to the IT environment.	IT
Applet deletion (ADELG)	The ADELG contains the security requirements for erasing installed applets from the card. It can also be used as a basis for any other application deletion requirements.	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 which do not support on-card static or dynamic verification of bytecodes, the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.	TOE
Card Lifecycle Management (LifeCycle)	The Lifecycle Group contains the minimal requirements that allow defining a policy for controlling access to card lifecycle management operations and for expressing card issuer security concerns. This group is within the scope of evaluation.	TOE
External Memory (EMG)	The EMG contains the requirements for a secure management of the external memory accessible to applet instances.	TOE

As a summary of this table, the scope of this TOE evaluation corresponds to the Open Configuration as defined in the Java Card Protection Profile.

Note that the code of the applets is not part of the code of the TOE, but just data managed by the TOE. Moreover, the scope of the ST does not include all the stages in the development cycle of a Java Card application described in Section 1.3.3. Applets are only considered in their CAP format, and the process of compiling the source code of an application and converting it into the CAP format does not regard the TOE or its environment. On the contrary, the process of verifying applications in its CAP format and loading it on the card is a crucial part of the TOE environment and plays an important role as a complement of the TSFs.

1.3.6 Smart Card Platform

The smart card platform (SCP) is composed of a micro-controller and hardware abstraction layer containing the cryptographic library (see Section 1.3.1). No separate operating system is present in this card. It provides memory management functions (such as separate interface to RAM and NVRAM), I/O functions that are compliant with ISO standards, transaction facilities, and secure implementation of cryptographic functions.

1.3.7 Native Applications

Apart from Java Card applications, the final product may contain native applications as well. Native applications are outside the scope of the TOE security functions (TSF), and they are usually written in the assembly language of the platform, hence their name. This term also designates software libraries providing services to other applications, including applets under the control of the TOE.

It is obvious that such native code presents a threat to the security of the TOE and to user applets.

Therefore, Java Card Protection Profile will require for native applications to be conformant with the TOE so as to ensure that they do not provide a means to circumvent or jeopardize the TSFs.

For the present products on J3E081_M64_DF, J3E081_M66_DF, J3E041_M64, J3E016_M64_DF, J3E041_M66 and J3E016_M66_DF, the certified hardware contains a native MIFARE DESFire application that belongs to the TOE. A TOE configured with the minor configuration option “MIFARE DESFire Emulation = A” does not provide an additional interface to the environment because the MIFARE DESFire application is logically disabled.

For J3E081_M64_DF, J3E081_M66_DF, J3E041_M64, J3E016_M64_DF, J3E041_M66 and J3E016_M66_DF the minor configurations configurations “MIFARE DESFire Emulation = D2”, “MIFARE DESFire Emulation = D4” and “MIFARE DESFire Emulation = D8” implement the contactless MIFARE DESFire OS and have access to 2KB, 4KB, or 8K EEPROM memory, respectively. Except native code which resides in the Secure BOX, the final product does not contain any other native applications according to JC PP. To completely securely separate the User OS and the MIFARE DESFire OS the smart card platform provides the so-called MIFARE firewall (see platform Security Targets **[10]**/**[9]**).

1.4 TOE Usage

Smart cards are mainly used as data carriers that are secure against forgery and tampering. More recent uses also propose them 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, usually 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 is inserted into a card reader. In some configurations of the TOE, the card reader may also be used to enlarge or restrict the set of applications that can be executed on the Java Card platform according to a well-defined card management policy.

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, and the balance of an electronic purse is highly sensitive with regard to arbitrary modification (because it represents real money).

So far, the most important 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) for digital mobile telephones.
- 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.

2. Conformance claims (ASE_CCL)

This chapter is divided into the following sections: “CC Conformance Claim", "Package claim", "PP claim", and “Conformance claim rationale”.

2.1 CC Conformance Claim

This Security Target claims to be conformant to version 3.1 of Common Criteria for Information Technology Security Evaluation according to

- “Common Criteria for Information Technology Security Evaluation, Part 1, Version 3.1, Revision 4, September 2012” [1]
- “Common Criteria for Information Technology Security Evaluation, Part 2, Version 3.1, Revision 4, September 2012”[2]
- “Common Criteria for Information Technology Security Evaluation, Part 3, Version 3.1, Revision 4, September 2012” [3]

The following methodology will be used for the evaluation.

- “Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 4, September 2012, CCMB-2012-09-004” [4]

This Security Target claims to be CC Part 2 extended and CC Part 3 conformant. The extended Security Functional Requirements are defined in Chapter 5.

2.2 Package claim

This Security Target claims conformance to the assurance package EAL 4 augmented. The augmentations to EAL4 are ALC_DVS.2, AVA_VAN.5, and ASE_TSS.2

2.3 PP claim

This Security Target claims conformance to the Protection Profile (PP)

"Java Card System - Open Configuration Protection Profile, Version 2.6, Certified by ANSSI, the French Certification Body April, 19th 2010" [5].

Since the Security Target claims conformance to this PP [5], the concepts are used in the same sense.

The TOE provides additional functionality, which is not covered in the PP [5].

2.4 Conformance claim rationale

2.4.1 TOE Type

The TOE type as stated in section 1.3.1 of this ST corresponds to the TOE type of the PP as stated in section 1.2 of [5] namely a Java Card platform, implementing the java card specification version 3.0.1.

2.4.2 SPD Statement

The SPD statement is presented in chapter 3 includes the threats as presented in the PP [5], but also includes a number of additional threats. These threats are:

- T.OS_OPERATE
- T.SEC_BOX_BORDER
- T.RND
- T.DF_DATA-MODIFICATION
- T.DF_IMPERSONATE
- T.DF_CLONING

The treat T.RND is taken from [6].

"This Protection Profile does not require formal compliance to a specific IC Protection Profile or a smart card OS Protection Profile but those IC and OS evaluated against [6] and [7] respectively, fully meet the objectives"

By adding these threats, the SPD is equivalent to the PP [5]

The threats T.OS_OPERATE and T.SEC_BOX_BORDER, are introduced to formulate the threats concerned with the secure box, which is identified as part of "additional native code" as defined in section 1.2 of the PP [5]. The threats T.DF_DATA-MODIFICATION, T.DF_IMPERSONATE, and T.DF_CLONING are introduced to formulate the threats concerned with the MIFARE DESFire functionality implemented in the underlying hardware. These threats are thus related to additional functionality, for which the PP offers the ability.

The SPD statement presented in chapter 3, copies the OSP from the PP [5], and adds OSP.PROCESS-TOE, this OSP is introduced for the pre-personalisation feature of the TOE. Furthermore, OSP.DESFire-Emulation is added for MIFARE DESFire functionality. Those two OSPs describe additional functionality for which the certified PP [5] offers the ability.

The SPD statement includes two of the three Assumptions from the PP [5]. The assumption A.Deletion is excluded. The card manager is part of the TOE and therefore the assumption is no longer relevant. Leaving out the assumption, makes the SPD in the [ST] more restrictive then the SPD in the PP [5]. The card manager is part of the TOE, is making sure that the Deletion of applets through the card manager is secure, instead of assuming that it is handled by the card manager in the environment of the TOE.

Besides the assumptions from the PP, are also five assumptions added:

- A.PROCESS-SEC-IC
- A.USE_DIAG
- A.USE_KEYS
- A.DF_SECURE-VALUES
- A.DF_TERMINAL-SUPPORT

The assumption A.PROCESS-SEC-IC, A.DF_SECURE-VALUE, and A.DF_TERMINAL-SUPPORT are taken from the underlying certified hardware platform [10], which is compliant to [6]. The assumptions A.USE_DIAG and A.USE_KEYS are included because the card manager is part of the TOE and no longer part of the environment. Adding these assumptions, this SPD is equivalent to the SPD in the PP [5].

2.4.3 Security Objectives Statement

The statement of security objectives in the ST presented in chapter 4 includes all security objectives as presented in the PP [5], but also includes a number of additional security objectives. These security objectives are:

- OT.SEC_BOX_FW
- OT.IDENTIFICATION
- OT.RND
- OT.MF_FW
- OT.DF_DATA-ACCESS
- OT.DF_AUTHENTICATION
- OT.DF_CONFIDENTIALITY
- OT.DF_TYPE-CONSISTENCY
- OT.DF_TRANSACTION

The security objectives OT.IDENTIFICATION, OT.RND, OT.MF_FW, OT.DF_DATA-ACCESS, OT.DF_AUTHENTICATION, OT.DF_CONFIDENTIALITY, OT.DF_TYPE-CONSISTENCY, and OT.DF_TRANSACTION are part of the security objectives of the Certified IC and Crypto Library, which is the component TOE ST from this composite product. Therefore the security objective statement is equivalent to the PP [5], for these security objectives. OT.IDENTIFICATION is also included for the pre-personalisation feature of the TOE, which is additional functionality the PP allows. The security objective OT.SEC_BOX_FW is the related to the introduction of the secure box, which is additional to the Java Card System functionality.

The statement of security objectives is therefore equivalent to the security objectives in the PP [5] to which conformance is claimed.

The ST introduces two additional security objectives for the environment besides part of the security objectives for the environment included from the PP [5]. The other security objectives for the environment are know, security objectives for the TOE.

- OE.USE_DIAG
- OE.USE_KEYS
- OE.PROCESS_SEC_IC

- OE.DF_Secure Values
- OE.DF_Terminal Support

The security objective for the environment OE.PROCESS_SEC_IC is from the platform (certified IC and crypto library) that is part from this composite product evaluation. Therefore the statement of security objectives for the environment is equivalent to the statement in the PP [5]. OE.USE_KEYS and OE.USE_DIAG are included because the card manager is part of the TOE and not a security objective for the environment as in PP [5]. OE.DF_Secure Values and OE.DF_Terminal Support are objectives to the environment to fulfill the functionality of the MIFARE DESFire functionality of the hardware. This is additional functionality which the PP [5] allows.

The statement of security objectives for the environment is therefore equivalent to the security objectives in the PP [5] to which conformance is claimed.

2.4.4 Security Requirements Statement

The statement of security functional requirements copies most SFRs as defined in the PP [5], with the exception from a number of options. For the copied set of SFRs the ST is considered equivalent to the statement of SFRs in the PP [5].

The TOE restricted remote access from the CAD to the services implemented by the applets on the card to none, and as a result FDP_ACF.1/JCRMI is modified. The remaining SFRs FDP_IFC.1/JCRMI, FDP_IFF.1/JCRMI, FMT_MSA.1/EXPORT, FMT_MSA.1/REM_REFS, FMT_MSA.3/JCRMI, FMT_SMF.1/JCRMI, FMT_REV.1/JCRMI, and FMT_SMR.1/JCRMI are not included in the ST. By removing the RMI, the statement of security functional requirements is more restrictive then the PP [5].

The ST includes the relevant SFRs from the platform ST [10] of this composite product. These SFRs are: FPT_FLS.1/SCP, FRU_FLT.2/SCP, FPT_PHP.3/SCP, FDP_ACC.1/SCP, FDP_ACF.1/SCP, FMT_MSA.3/SCP and FAU_SAS.1/SCP. For this set of SFRs, the ST is considered equivalent to the statement of SFRs in the PP [5], because it realizes a [6] conformant platform, which fully meets the objectives as stated in section 1.2 of the PP [5].

The set of SFRs that define the functionality of the MIFARE DESFire, realizes additional security functionality making the security requirements statement equivalent to the PP [5]. This set of SFRs comprise FMT_SMR.1[DESFire], FDP_ACC.1[DESFire], FDP_ACF.1[DESFire], FMT_MSA.3[DESFire], FMT_MSA.1[DESFire], FMT_SMF.1[DESFire], FDP_ITC.2[DESFire], FCS_CKM.4[DESFire], FMT_MTD.1[DESFire], FCS_COP.1[DESFire_HW_DES], FCS_COP.1[DESFire_HW_AES], FIA_UID.2[DESFire], FIA_UAU.2[DESFire], FIA_UAU.5[DESFire], FTP_TRP.1[DESFire], FPT_RPL.1[DESFire], FPT_TDC.1[DESFire], and FDP_ROL.1[DESFire]

The set of SFRs that define the Secure Box, realize additional security functionality making the security requirements statement equivalent to the PP [5]. This set of SFRs comprise FDP_ACC.2/SecureBox , FDP_ACF.1/SecureBox , FMT_MSA.3/SecureBox, FMT_MSA.1/SecureBox and FMT_SMF.1/SecureBox.

The set of SFRs that are included because of inclusion of the Card Manager and a pre-personalisation feature in the TOE add the following SFRs: FDP_ACC.1/LifeCycle, FDP_ACF.1/ LifeCycle, FMT_MSA.1/ LifeCycle,FMT_MSA.3/ LifeCycle, FMT_SMR.1/ LifeCycle and FTP_ITC.1/LifeCycle

The SFRs FIA_AFL.1/PIN, FCS_RNG.1, FCS_RNG.1/RNG2 and FPT_EMSEC.1,add functionality to the TOE making the statement of security requirements more restrictive then the PP [5].

3. Security problem definition (ASE_SPD)

3.1 Introduction

This chapter describes the security problem to be addressed by the TOE and the operational environment of the TOE. The security problem is described by threats for the assets. The assets are described in Section 3.2, whereas threats are described in section 3.3. Organisational Security Policies are given in Section 3.4 and the Assumptions are made in Section 3.5. Finally Section 3.6 defines some security aspects. Security aspects are intended to define the main security issues that are to be addressed in the PP and this ST, in a CC-independent way. They can be instantiated as assumptions, threats, and objectives.

The description is based on [5] and supplemented by the description of [6].

3.2 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 have to be protected, some in terms of confidentiality and some in terms of integrity or both integrity and confidentiality. These assets are concerned by the threats on the TOE and include

a. TOE including NOS code,

b. TSF data, as initialization data, configuration data, cryptographic keys, random numbers for key generation, and all data used by the TOE to execute its security functions. This includes also configuration of hardware specific security features.

c. User Data, as application code (applets), specific sensitive application values, as well as application specific PIN and authentication data.

d. MIFARE DESFire Operating System Data, as initialization data, configuration data, cryptographic keys, random numbers for key generation, and all data used by the d. MIFARE DESFire Operating System to execute its security functions. This includes also configuration of hardware specific security features.

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) data created by and for the MIFARE DESFire Operating System. The definition is taken from section 5.1 of [5].

3.2.1 User Data

D.APP_CODE	The code of the applets and libraries loaded on the card. To be protected from unauthorized modification.
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.PIN	Any end-user's PIN. To be protected from unauthorized disclosure and modification.
D.APP_KEYs	Cryptographic keys owned by the applets. To be protected from unauthorized disclosure and modification. TSF Data
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 JCVM, 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 monopolization and unauthorized disclosure or modification.
D.SEC_DATA	The runtime security data of the JCRE, 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.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.ADMIN_CONF_DATA	Private data of the System accessible via the root applet if authenticated with a admin key, like quality parameters for key generation, memory layout settings, transport key.
D.PERSO_CONF_DATA	Private data of the System accessible via the root applet if authenticated with a transport or admin key, like protocol parameters, compliance settings.

3.2.2 MIFARE DESFire Data

D.DF_DATA	Keys, Files and Values controlled by the MIFARE DESFire Emulation.

	To be protected from unauthorized disclosure and modification.
D.DF_CODE	MIFARE DESFire Emulation, stored and in operation To be protected from unauthorized disclosure and modification.

3.3 Threats

This section introduces the threats to the assets against which specific protection within the TOE or its environment is required. It is assumed that all attackers have high level of expertise, opportunity and resources. General threats for smart card native operating systems were defined and supplemented by Java Card specific threats from [5]. In addition also the threats for the MIFARE DESFire Emulation are taken from [10]. Only threats on TOE information during phase 7 are considered. They are summarized in the following table:

Table 9. Threats

Name	Source	Refined?
T.OS_OPERATE	-	-
T.SEC_BOX_BORDER	-	-
T.RND	[6]	no
T.CONFID-APPLI-DATA	[5]	no
T.CONFID-JCS-CODE	[5]	no
T.CONFID-JCS-DATA	[5]	no
T.INTEG-APPLI-CODE	[5]	no
T.INTEG-APPLI-CODE.LOAD	[5]	no
T.INTEG-APPLI-DATA	[5]	no
T.INTEG-APPLI-DATA.LOAD	[5]	no
T.INTEG-JCS-CODE	[5]	no
T.INTEG-JCS-DATA	[5]	no
T.SID.1	[5]	no
T.SID.2	[5]	no
T.EXE-CODE.1	[5]	no
T.EXE-CODE.2	[5]	no
T.EXE-CODE-REMOTE	[5]	no
T.NATIVE	[5]	no
T.RESOURCES	[5]	no
T.DELETION	[5]	no

Name	Source	Refined?
T.INSTALL	[5]	no
T.OBJ-DELETION	[5]	no
T.PHYSICAL	[5]	yes[3]
T.DF_DATA-MODIFICATION	[10]	no
T.DF_IMPERSONATE	[10]	no
T.DF_CLONING	[10]	no

3.3.1 Threats not contained in [5] or [10]

The TOE is required to counter the threats described hereafter; a threat agent wishes to abuse the assets either by functional attacks or by environmental manipulation, by specific hardware manipulation, by a combination of hardware and software manipulations or by any other type of attacks.

Threats have to be split in

- Threats against which specific protection within the TOE is required,
- Threats against which specific protection within the environment is required.

3.3.1.1 Unauthorized full or partial Cloning of the TOE

The cloning of the functional behavior of the Smart Card on its ISO command interface is the highest-level security concern in the application context. The cloning of that functional behavior requires:

- To develop a functional equivalent of the Smart Card Native Operating System and its applications, to disclose, to interpret and employ the secret User Data stored in the TOE, and
- To develop and build a functional equivalent of the Smart Card using the input from the previous steps.

The Native Operating System must ensure that especially the critical User Data are stored and processed in a secure way but also ensures that critical User Data are treated as required in the application context. In addition, the personalization process supported by the Smart Card Native Operating System (and by the Smart Card Integrated Circuit in addition) must be secure.

This last step is beyond the scope of this Security Target. As a result, the threat “cloning of the functional behavior of the Smart Card on its ISO command interface” is averted by the combination of measures, which split into those being evaluated according to this Security Target and the corresponding personalization process. Therefore, functional cloning is indirectly covered by the threats described below.

3.3.1.2 Threats on TOE operational environment

The TOE is intended to protect itself against the following threats

[3] Refinement to cover additional aspects of O.SCP.IC not contained in [5].

- Manipulation of User Data and of the Smart Card Native Operating System (while being executed/processed and while being stored in the TOE's memories) and
- Disclosure of User Data and of the Smart Card NOS (while being processed and while being stored in the TOE's memories).

The TOE's countermeasures are designed to avert the threats described below. Nevertheless, they may be effective in earlier phases (phases 4 to 6).

Though the Native Operating System (normally stored in the ROM) will in many cases not contain secret data or algorithms, it must be protected from being disclosed, since for instance knowledge of specific implementation details may assist an attacker. In many cases critical User Data and NOS configuration data (TSF data) will be stored in the EEPROM.

3.3.1.3 Software Threats

The most basic function of the Native Operating System is to provide data storage and retrieval functions with a variety of access control mechanisms which can be configured to suit the embedded application(s) context requirements.

Each authorized role has certain specified privileges which allow access only to selected portions of the TOE and the information it contains. Access beyond those specified privileges could result in exposure of assets. On another hand, an attacker may gain access to sensitive data without having permission from the entity that owns or is responsible for the information or resources.

T.OS_OPERATE — Modification of the correct NOS behavior by unauthorized use of TOE or use of incorrect or unauthorized instructions or commands or sequence of commands, in order to obtain an unauthorized execution of the TOE code.
An attacker may cause a malfunction of TSF or of the Smart Card embedded NOS in order to (1) bypass the security mechanisms (i.e. authentication or access control mechanisms) or (2) obtain unexpected result from the embedded NOS behavior
Different kind of attack path may be used as:

- Applying incorrect unexpected or unauthorized instructions, commands or command sequences,
- Provoking insecure state by insertion of interrupt (reset), premature termination of transaction or communication between IC and the reading device

Complementary note — Any implementation flaw in the NOS itself can be exploited with this attack path to lead to an unsecured state of the state machine of the NOS.
The attacker uses the available interfaces of the TOE.
A user could have certain specified privileges that allow loading of selected programs. Unauthorized programs, if allowed to be loaded, may include either the execution of legitimate programs not intended for use during normal operation (such as patches, filters, Trojan horses, etc.) or the unauthorized loading of programs specifically targeted at penetration or modification of the security functions. Attempts to generate a non-secure state in the Smart Card may also be

	made through premature termination of transactions or communications between the IC and the card reading device, by insertion of interrupts, or by selecting related applications that may leave files open.
T.SEC_BOX_BORDER	An attacker may try to use malicious code placed in the Secure Box to modify the correct behavior of the NOS. With the aim to (1) disclose the Java Card System code, (2) disclose or alter Applet code, disclose or alter Java Card System data, or disclose or alter Applet data.

3.3.1.4 Threat on Random Numbers

The following threat was taken over from [6]:

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. 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.

3.3.2 Threats from [5]

The following threats specific for the Java Card functionality were taken from [5].

3.3.2.1 Confidentiality

T.CONFID-APPLI-DATA	The attacker executes an application to disclose data belonging to another application.. See #.CONFID-APPLI-DATA (p. 35) 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 (p. 35) 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 (p. 35) for details. Directly threatened asset(s): D.API_DATA, D.SEC_DATA, D.JCS_DATA D.JCS_KEYs and D.CRYPTO.

3.3.2.2 Integrity

T.INTEG-APPLI-CODE	The attacker executes an application to alter (part of) its own or another application's code. See #.INTEG-APPLI-CODE

(p. 35) 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 (p. 35) 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 (p. 35) 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 (p. 35) for details.
Directly threatened asset(s): 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 (p. 35) 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 (p. 35) for details.
Directly threatened asset(s): D.API_DATA, D.SEC_DATA, D.JCS_DATA, D.JCS_KEYs 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.

3.3.2.3 Identity Usurpation

T.SID.1 An applet impersonates another application, or even the JCRE, in order to gain illegal access to some resources of the card or with respect to the end user or the terminal. See #.SID (p. 38) 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 and D.APP_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 (p. 38) 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).

3.3.2.4 Unauthorized Execution

T.EXE-CODE.1	An applet performs an unauthorized execution of a method. See #.EXE-JCS-CODE (p. 36) and #.EXE-APPLI-CODE (p. 36) for details. Directly threatened asset(s): D.APP_CODE.
T.EXE-CODE.2	An applet performs an unauthorized execution of a method fragment or arbitrary data. See #.EXE-JCS-CODE (p. 36) and #.EXE-APPLI-CODE (p. 36) for details. Directly threatened asset(s): D.APP_CODE.
T.EXE-CODE-REMOTE	The attacker performs an unauthorized remote execution of a method from the CAD. See #.EXE-APPLI-CODE (p. 36) 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 (p. 36) for details. Directly threatened asset(s): D.JCS_DATA.

3.3.2.5 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 (p. 39) for details. Directly threatened asset(s): D.JCS_DATA.

3.3.2.6 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 (p. 38) 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 (p. 38) 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).

3.3.2.7 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 (p. 38) for further details. Directly threatened asset(s): D.APP_C_DATA, D.APP_I_DATA and D.APP_KEYs.

3.3.2.8 Miscellaneous

T.PHYSICAL — The attacker discloses or modifies the design of the TOE, its sensitive data **(TSF and User Data)** or application code **or disables security features of the TOE** 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.

Note: This threat from [5] was refined to cover additional aspects not contained in [5].

3.3.3 Threats from [10]

T.DF_DATA-MODIFICATION — Unauthorized modification of keys, files and values maintained by the MIFARE DESFire Emulation.
Keys, files and values maintained by the MIFARE DESFire Emulation are processed and stored by the TOE. They may be modified by unauthorized subjects. This threat applies to the processing of modified commands received by the TOE, it is not concerned with verification of authenticity.

T.DF_IMPERSONATE — Impersonating authorized users during the authentication process of the MIFARE DESFire Emulation.
An unauthorized subject may try to impersonate an authorized subject during the authentication sequence of the MIFARE DESFire Emulation, e.g. by a man-in-the middle or replay attack.

T.DF_CLONING — Cloning using keys, files and values maintained by the MIFARE DESFire Emulation
Keys, files and values maintained by the MIFARE DESFire Emulation stored on the TOE may be read out by an unauthorized subject in order to create a duplicate.

3.4 Organisational security policies (OSPs)

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 (p.36) for details.

OSP.PROCESS-TOE — An accurate identification must be established for the TOE. This requires that each instantiation of the TOE carries this identification.

Note: The IC Developer / Manufacturer must apply the policy "Protection during TOE Development and Production (OSP.PROCESS-TOE)" as specified above.

In addition the MIFARE DESFire Emulation as part of the hardware platform provides the following security functionality "P.DESFire-Emulation". The Security IC Embedded Software can call the MIFARE DESFire Emulation which implements this security policy. It is not mandatory for the Security IC Embedded Software to call the MIFARE DESFire Emulation because the policy described above is independent of the MIFARE DESFire Emulation. However if the TOE shall emulate the MIFARE DESFire functionality the Java Card Systems must call the MIFARE DESFire Emulation. Therefore the I the additional policies are defined as specified below.

OSP.DESFire-Emulation The MIFARE DESFire Emulation provides the following specific security components:

- Confidentiality during communication provides the possibility to protect selected data elements from eavesdropping during contactless communication.
- Integrity during communication provides the possibility to protect the contactless communication from modification or injections. This includes especially the possibility to detect replay or man-in-the-middle attacks within a session.
- Transaction mechanism provides the possibility to combine a number of data modification operations in one transaction, so that either all operations or no operation at all is performed.

3.5 Assumptions

This section is partly taken from [5] and 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" ([21], §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.

The following two assumptions are outside the control of the MIFARE DESFire Emulation. These assumptions must be implemented to support the security functionality of the MIFARE DESFire Emulation.

A.DF_SECURE-VALUES Usage of secure values.
Only confidential and secure keys shall be used to set up the authentication and access rights for the MIFARE DESFire Emulation. These values are generated outside the TOE. They must be protected during generation, management outside the TOE and downloaded to the TOE.

A.DF_TERMINAL-SUPPORT Terminal support to ensure integrity and confidentiality. The terminal verifies information sent by the TOE in order to ensure integrity and confidentiality of the communication.

In addition to the assumptions taken from [5] an additional assumption is made which is describing the protection during packaging, finishing, and personalization.

A.USE_DIAG It is assumed that the operational environment supports and uses the secure communication protocols offered by TOE.

A.USE_KEYS It is assumed that the keys which are stored outside the TOE and which are used for secure communication and authentication between Smart Card and terminals are protected for confidentiality and integrity in their own storage environment.

Note: This is to assume that the keys used in terminals or systems are correctly protected for confidentiality and integrity in their own environment, as the disclosure of such information which is shared with the TOE but is not under the TOE control, may compromise the security of the TOE.

A.PPROCESS-SEC-IC It is assumed that security procedures are used after delivery of the TOE by the TOE Manufacturer up to delivery to the endconsumer to maintain confidentiality and integrity of the TOE and of its manufacturing and test data (to prevent any possible copy, modification, retention, theft or unauthorised use). This means that the Phases after TOE Delivery (refer to Section 1.3.3) are assumed to be protected appropriately.The assets to be protected are:

- The information and material produced and/or processed by the Security IC Embedded Software Developer in Phase 1 and by the Composite Product Manufacturer can be grouped as follows:

- the Security IC Embedded Software including specifications, implementation and related documentation,

- pre-personalisation and personalisation data including specifications of formats and memory areas, test related data,

- the User Data and related documentation, and

- material for software development support

as long as they are not under the control of the TOE Manufacturer. Details must be defined in the Protection Profile or Security Target for the evaluation of the Security IC Embedded Software and/or Security IC.

3.6 Security Aspects

This section is partly taken from [5].

Security aspects are intended to define the main security issues that are to be addressed in the PP and this ST, 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 and are referenced in their definition. For instance, the security aspect #.NATIVE is instantiated in assumption A.NATIVE and objectives OE.NATIVE, and the security aspect #.FIREWALL is instantiated in the objective OT.FIREWALL.

The following sections present several security aspects from [5] that are relevant for this ST.

3.6.1 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. 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 runtime environment, the virtual machine and the internal data of Java Card API classes as well.

3.6.2 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. If the configuration allows 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. If the configuration allows 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 runtime environment, the virtual machine and the internal data of Java Card API classes as well.

3.6.3 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 visibility rules provided by the public/private access modifiers of the Java programming language ([14],§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.

#.EXE-JCS-CODE — Java Card System (byte)code must be protected against unauthorized execution. Java Card System (byte)code includes any code of the JCRE or API. This concerns **(1)** invoking a method outside the scope of the visibility rules provided by the public/private access modifiers of the Java programming language ([14],§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 Java Card System shall ensure controlled sharing of class instances4, and isolation of their data and code between packages (that is, controlled execution contexts). (1) An applet shall neither read, write nor compare a piece of data belonging to an applet that is not in the same context, nor 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 TOE Scope Control (TSC), it must be secured so as to not provide ways to bypass the TSFs. No untrusted native code may reside on the card. Loading of native code, which is as well outside the TSC, 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.

3.6.3.1 Bytecode Verification

#.VERIFICATION — All 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.

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

3.6.3.2 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 [21], [13]). The actual set of checks performed by the verifier is implementation-dependent, but shall at least enforce all the "must clauses" imposed in [21] on the bytecodes and the correctness of the CAP files' format.

As most of the actual JCVMs 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.

Note: In the present case, bytecode verification is performed before loading.

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 [21], §4.4), second, that the export files used to check and link the loaded applet have the corresponding correct counterpart on the card.

3.6.3.3 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.

Alternatively, the card itself may include a verifier and perform the checks prior to the effective installation of the applet or provide means for the bytecodes to be verified dynamically.

Note: In the present case, bytecode verification is performed before loading.

3.6.3.4 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.

3.6.4 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	Installation of a package or an applet is secure. **(1)** The TOE must be able to return to a safe and consistent state should the installation fail or be cancelled (whatever the reasons). **(2)** Installing an application 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 a secure atomic operation, and 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 2.2.1). 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	Deallocation of objects must be secure. **(1)** It 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	Deletion of applets must be secure. **(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 TSPs. 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 TSPs.
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).

3.6.5 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:	In the present case 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).

4. Security objectives for the TOE

The Security Objectives for the TOE are summarized in the following table:

Table 10. Security Objectives for the TOE

Name	Source	Refined?

Name	Source	Refined?
OT.SEC_BOX_FW	-	-
OT.IDENTIFICATION	-	-
OT.SID	[5]	no
OT.FIREWALL	[5]	no
OT.GLOBAL_ARRAYS_CONFID	[5]	no
OT.GLOBAL_ARRAYS_INTEG	[5]	no
OT.NATIVE	[5]	no
OT.OPERATE	[5]	no
OT.REALLOCATION	[5]	no
OT.RESOURCES	[5]	no
OT.ALARM	[5]	no
OT.CIPHER	[5]	no
OT.KEY-MNGT	[5]	no
OT.PIN-MNGT	[5]	no
OT.REMOTE	[5]	no
OT.TRANSACTION	[5]	no
OT.OBJ-DELETION	[5]	no
OT.DELETION	[5]	no
OT.LOAD	[5]	no
OT.INSTALL	[5]	no
OT.CARD-MANAGEMENT	[5]	no (*)
OT.SCP.IC	[5]	no (*)
OT.SCP.RECOVERY	[5]	no (*)
OT.SCP.SUPPORT	[5]	no (*)
OT.EXT-MEM	[5]	no
OT.RND	[6]	no
OT.MF_FW	[10]	no
OT.DF_DATA-ACCESS	[10]	no
OT.DF_AUTHENTICATION	[10]	no
OT.DF_CONFIDENTIALITY	[10]	no

Name	Source	Refined?
OT.DF_TYPE-CONSISTENCY	[10]	no
OT.DF_TRANSACTION	[10]	no

(*) These Security Objectives for the environment of [5] are Security Objectives for the TOE in the present evaluation. Therefore, the label changed (OT.XYZ instead of OE.XYZ) but not the content (no refinement).

4.1.1 Security Objectives for the TOE not contained in [5] or [10]

The security objectives of the TOE must cover the following ***aspects***:

- Maintain the integrity of User Data and of the Smart Card Native Operating System (when being executed/processed and when being stored in the TOE's memories) and
- Maintain the confidentiality of User Data and of the Smart Card Native Operating System (when being processed and when being stored in the TOE's memories), as well as
- Provide access control to execution of the TOE code
- Ensure correct operation of the code and maintain the TOE in a secure state

OT.SEC_BOX_FW	The TOE shall provide separation between the Secure Box native code and the Java Card System. The separation shall comprise software execution and data access.
OT.IDENTIFICATION	The TOE must provide means to store Initialization Data and Pre-personalization Data in its non-volatile memory. The Initialization Data (or parts of them) are used for TOE identification.
OT.MF_FW	The TOE shall provide separation between the "MIFARE DESFire Operating System" IC Dedicated Support Software and the Smartcard Embedded Software. The separation shall comprise software execution and data access.
OT.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.

4.1.2 Security Objectives for the TOE from [5]

4.1.2.1 Identification

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

4.1.2.2 Execution

OT.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 (p 36) for details.
OT.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.
OT.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.
OT.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 (p.36) for details.
OT.OPERATE	The TOE must ensure continued correct operation of its security functions. Especially, the TOE must prevent the unauthorized use of TOE or use of incorrect or unauthorized instructions or commands or sequence of commands. See #.OPERATE (p. 39) for details.
OT.REALLOCATION	The TOE shall ensure that the re-allocation of a memory block for the runtime areas of the JCVM does not disclose any information that was previously stored in that block.
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 Java Card Virtual Machine Specification [21].
OT.RESOURCES	The TOE shall control the availability of resources for the applications. See #.RESOURCES (p 39) for details.

4.1.2.3 Services

OT.ALARM	The TOE shall provide appropriate feedback information upon detection of a potential security violation. See #.ALARM (p. 39) for details.
OT.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 (p. 39) for details.
OT.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 #.KEY-MNGT (p. 39).
OT.PIN-MNGT	The TOE shall provide a means to securely manage PIN objects. See #.PIN-MNGT (p. 40) 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.*
Note:	For this Java Card such libraries do not exist. All necessary functionality is implemented by the TOE.
OT.REMOTE	The TOE shall provide restricted remote access from the CAD to the services implemented by the applets on the card. This particularly concerns the Java Card RMI services introduced in version 2.2.x of the Java Card platform.
OT.TRANSACTION	The TOE must provide a means to execute a set of operations atomically. See #.TRANSACTION (p. 40) for details.
Note:	OT.KEY-MNGT, OT.PIN-MNGT, OT.TRANSACTION and OT.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.

4.1.2.4 Object Deletion

OT.OBJ-DELETION	The TOE shall ensure the object deletion shall not break references to objects. See #..OBJ-DELETION (p. 38) for further details.

4.1.2.5 Applet Management

OT.DELETION	The TOE shall ensure that both applet and package deletion perform as expected. See #.DELETION for details.
OT.LOAD	The TOE shall ensure that the loading of a package into the card is safe. *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.*
OT.INSTALL	The TOE shall ensure that the installation of an applet performs as expected (See #.INSTALL for details).

4.1.2.6 Card Management

The TOE Security Objective for the card manager is a Security Objective for the environment in [5]. In the present case the card manager belongs to the TOE and the corresponding Security Objective is listed here.

OT.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.

Note: The Security Objective from [5] for the environment OE.CARD-MANAGEMENT is listed as TOE security objective for the TOE in section 4.1.2.6 as the Card Manager belongs to the TOE for this evaluation.

4.1.2.7 Smart Card Platform

These TOE Security Objectives for the smart card platform are Security Objectives for the environment in [5]. In the present case the certified smart card platform belongs to the TOE and the corresponding Security Objectives are listed here.

OT.SCP.IC The SCP shall provide all IC security features against physical attacks. See #.SCP.7 (p.40).

OT.SCP.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 (#.SCP.1). (p.40).:

OT.SCP.SUPPORT The SCP shall support the TSFs of the TOE. This security objective for the environment refers to the security aspects 2, 3, 4 and 5 of #.SCP (p.40).:

Note: The Security Objectives from [5] for the environment OE.SCP.RECOVERY, OE.SCP.SUPPORT, and OT.SCP.IC are listed as TOE security objectives for the TOE in section 4.1.2.7 as the smart card platform belong to the TOE for this evaluation.

4.1.2.8 EMG Extended Memory

This TOE Security Objective for the extended memory feature is a objective described in Appendix A of the PP [5] and comes with the compliance to Java Card 3.0.1.

OT.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).

4.1.3 Security Objectives for the TOE from [10]

The security objectives of the MIFARE DESFire Emulation can only be provided if the MIFARE DESFire Emulation is called by JCOP. The MIFARE DESFire Emulation is part of the TOE and provides the following security objectives:

OT.DF_DATA-ACCESS Access Control to DESFire Data
The TOE must provide an access control mechanism for data stored by the MIFARE DESFire Emulation. The access control mechanism shall apply to read, modify, create and delete operations for data elements and to reading and modifying security attributes as well as authentication data. It shall be possible to limit the right to perform a specific operation to a specific user. The security attributes (keys) used for authentication shall never be output.

OT.DF_AUTHENTICATION Authentication
The MIFARE DESFire Emulation as part of the TOE must provide an authentication mechanism in order to be able to authenticate authorized users. The authentication mechanism shall be limited to the MIFARE DESFire Emulation and shall be resistant against replay and man-in-the-middle attacks.

OT.DF_CONFIDENTIALITY Confidential Communication
The TOE must be able to protect the communication of the MIFARE DESFire Emulation by encryption. This shall be implemented by security attributes of the DESFire data element that enforce encrypted communication of the MIFARE DESFire Emulation for the respective data element.
During DESFire operation the TOE shall also provide the possibility to detect replay or man-in-the-middle attacks within a session. This shall be implemented by checking verification data sent by the terminal and providing verification data to the terminal.

OT.DF_TYPE-CONSISTENCY Data type consistency
The TOE must provide a consistent handling of the data types (files and values) of the MIFARE DESFire Emulation. This comprises over- and underflow checking for values, for data file sizes and for record handling.

OT.DF_TRANSACTION Transaction mechanism
The TOE must be able to provide a transaction mechanism that allows to update multiple data elements of the MIFARE DESFire Emulation either all in common or none of them.

4.2 Security objectives for the operational environment

The Security Objectives for the operational environment are summarized in the following table:

Table 11. Security Objectives for the operational environment

Name	Source	Refined?
OE.USE_DIAG	-	-
OE.USE_KEYS	-	-
OE.PROCESS_SEC_IC	-	-
OE.VERIFICATION	[5]	no
OE.APPLET	[5]	no
OE.DF_Secure Values	[10]	no
OE.DF_Terminal Support	[10]	no

4.2.1 Security Objectives for the operational environment not contained in [5] or [10]

4.2.1.1 Objectives on Phase 7

OE.USE_DIAG — Secure TOE communication protocols shall be supported and used by the environment.

OE.USE_KEYS — During the TOE usage, the terminal or system in interaction with the TOE, shall ensure the protection (integrity and confidentiality) of their own keys by operational means and/or procedures.

<u>Note:</u> Objectives for the TOE environment are usually not satisfied by the TOE Security Functional Requirements.

The TOE development and manufacturing environment (phases 1 to 3) is in the scope of this ST. These phases are under the TOE developer scope of control. Therefore, the objectives for the environment related to phase 1 to 3 are covered by Assurance measures, which are materialized by documents, process and procedures evaluated through the TOE evaluation process.

The `product usage phases` (phase 4 to 7) are not in the scope of the evaluation. During these phases, the TOE is no more under the developer control. In this environment, the TOE protects itself with its own Security functions. But some additional usage recommendation must also be followed in order to ensure that the TOE is correctly and securely handled, and that shall be not damaged or compromised. This ST assumes (A.USE_DIAG, A.USE_KEYS) that users handle securely the TOE and related Objectives for the environment are defined (OE.USE_DIAG, OE.USE_KEYS).

OE.PROCESS_SEC_IC — Protection during composite product manufacturing

Security procedures shall be used after TOE Delivery up to delivery to the end-consumer to maintain confidentiality and integrity of the TOE and of its manufacturing and test data (to prevent any possible copy, modification, retention, theft or unauthorised use). This means that Phases after TOE Delivery up to the end of Phase 6 (refer to Section 1.3.3) must be protected appropriately.

4.2.2 Security Objectives for the operational environment from [5]

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 (p.36) for details.

4.2.3 Security Objectives for the operational environment from [10]

OE.DF_Secure Values	Generation of secure values The environment shall generate confidential and secure keys for authentication purpose of the MIFARE DESFire Emulation. These values are generated outside the TOE and they are downloaded to the TOE during the personalization or usage in phase 5 to 7
OE.DF_Terminal Support	Terminal support to ensure integrity and confidentiality The terminal shall verify information sent by the MIFARE DESFire Emulation in order to ensure integrity and confidentiality of the communication. This involves checking of MAC values, verification of redundancy information according to the cryptographic protocol and secure closing of the communication session.

4.3 Security Objectives Rationale

In this section it is proven that the security objectives described in section 4 can be traced for all aspects identified in the TOE-security environment and that they are suited to cover them.

At least one security objective results from each assumption, OSP, and each threat. At least one threat, one OSP or assumption exists for each security objective.

Table 12. Assignment: threats / OSP – security objectives for the TOE

	OT.SEC_BOX_FW	OT.SID	OT.FIREWALL	OT.GLOBAL_ARRAYS_CONFID	OT.GLOBAL_ARRAYS_INTEG	OT.NATIVE	OT.OPERATE	OT.REALLOCATION	OT.RESOURCES	OT.ALARM	OT.CIPHER	OT.KEY-MNGT	OT.PIN-MNGT	OT.REMOTE	OT.TRANSACTION	OT.OBJ-DELETION	OT.DELETION	OT.LOAD	OT.INSTALL	OT.CARD-MANAGEMENT	OT.SCP.IC	OT.SCP.RECOVERY	OT.SCP.SUPPORT	OT.EXT-MEM	OT.IDENTIFICATION	OT.RND	OT.MF_FW
T.OS_OPERATE	x																										x
T.SEC_BOX_BORDER	x																										
T.RND																										x	
T.CONFID-APPLI-DATA		x	x	x			x	x		x	x	x	x		x					x		x	x	x			
T.CONFID-JCS-CODE						x														x				x			
T.CONFID-JCS-DATA		x	x				x			x										x		x	x	x			
T.INTEG-APPLI-CODE						x														x				x			
T.INTEG-APPLI-CODE.LOAD																		x		x							
T.INTEG-APPLI-DATA		x	x		x		x	x		x	x	x	x		x					x		x	x				
T.INTEG-APPLI-DATA.LOAD																		x		x							
T.INTEG-JCS-CODE						x														x				x			
T.INTEG-JCS-DATA		x	x				x			x										x		x	x	x			
T.SID.1		x	x																	x							
T.SID.2		x	x				x															x	x				
T.EXE-CODE-REMOTE														x													
T.NATIVE						x																					
T.RESOURCES							x	x														x	x				
T.DELETION																	x			x							
T.INSTALL																		x	x	x							
T.OBJ-DELETION																x											
T.PHYSICAL																					x						
OSP.PROCESS-TOE																									x		

Table 13. Assignment: threats / OSP – security objectives for the TOE according to the DESFire Emulation

	OT.DF_DATA-ACCESS	OT.DF_AUTHENTICATION	OT.DF_CONFIDENTIALITY	OT.DF_TYPE-CONSISTENCY	OT.DF_TRANSACTION
T.DF_DATA_MODIFICATION	x			x	
T.DF_IMPERSONATE		x			
T.DF_CLONING	x	x			
OSP.DESFire-Emulation			x	x	x

Table 14. Assignment: threats / assumptions / OSP – security objectives for the environment

	OE.USE_DIAG	OE.USE_KEY	OE.PROCESS_SEC_IC	OE.VERIFICATION	OE.APPLET
T.CONFID-APPLI-DATA				x	
T.CONFID-JCS-CODE				x	
T.CONFID-JCS-DATA				x	
T.INTEG-APPLI-CODE				x	
T.INTEG-APPLI-DATA				x	
T.INTEG-JCS-CODE				x	
T.INTEG-JCS-DATA				x	
T.EXE-CODE.1				x	
T.EXE-CODE.2				x	
T.NATIVE				x	x
A.USE_DIAG	x				

	OE.USE_DIAG	OE.USE_KEY	OE.PROCESS_SEC_IC	OE.VERIFICATION	OE.APPLET
A.USE_KEY		x			
A.PROCESS_SEC_IC			x		
A.APPLET					x
A.VERIFICATION				x	
OSP.VERIFICATION				x	

Table 15. Assignment: threats / assumptions / OSP – security objectives for the environment according to the DESFire Emulation

	OE.DF_SECURE-VALUES	OE.DF_TERMINAL-SUPPORT
A.DF_SECURE-VALUES	x	
A.DF_TERMINAL-SUPPORT		x

4.3.1 Security Objectives Rationale from [5]

The following chapters have been taken from [5] without modifications.

4.3.1.1 Threats

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 (OT.FIREWALL) objective. It relies in its turn on the correct identification of applets stated in (OT.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (OT.OPERATE) objective. As the firewall is a software tool automating critical controls, the objective OT.ALARM asks for it to provide clear warning and error messages, so that the appropriate countermeasure can be taken. The objectives OT.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 OT.SCP.RECOVERY and OT.SCP.SUPPORT are intended to support the OT.OPERATE and OT.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 (OT.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 (OT.KEY-MNGT, OT.PIN-MNGT, OT.TRANSACTION). If the PIN class of the Java Card API is used, the objective (OT.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 OT.GLOBAL_ARRAYS_CONFID. Furthermore, any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the OT.REALLOCATION objective. That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused. Finally, the objective OT.EXT-MEM provides access control for external memory and therefore also contributes to counter this threat.

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 OT.NATIVE, so no application can be run to disclose a piece of code. The (#.VERIFICATION) security aspect is addressed in this PP by the objective for the environment OE.VERIFICATION. The objectives OT.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively. Finally, the objective OT.EXT-MEM provides access control for external memory and therefore also contributes to counter this threat.

T.CONFID-JCS-DATA This threat is covered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (OT.FIREWALL) security objective. This latter objective also relies in its turn on the correct identification of applets stated in (OT.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (OT.OPERATE) objective. As the firewall is a software tool automating critical controls, the objective OT.ALARM asks for it to provide clear warning and error messages, so that the appropriate countermeasure can be taken. The objectives OT.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 OT.SCP.RECOVERY and OT.SCP.SUPPORT are intended to support the OT.OPERATE and OT.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter. Finally, the objective OT.EXT-MEM provides access control for external memory and therefore also contributes to counter this threat.

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 OT.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 OT.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively. Finally, the objective OT.EXT-MEM provides access control for external memory and therefore also contributes to counter this threat.

T.INTEG-APPLI-CODE.LOAD This threat is countered by the security objective OT.LOAD which ensures that the loading of packages is done securely and thus preserves the integrity of packages code. By controlling the access to card management functions such as the installation, update or deletion of applets the objective OT.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 (OT.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (OT.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (OT.OPERATE) objective. As the firewall is a software tool automating critical controls, the objective OT.ALARM asks for it to provide clear warning and error messages, so that the appropriate countermeasure can be taken. The objectives OT.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 OT.SCP.RECOVERY and OT.SCP.SUPPORT are intended to support the OT.OPERATE and OT.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 (OT.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 (OT.KEY-MNGT, OT.PIN-MNGT, OT.TRANSACTION). If the PIN class of the Java Card API is used, the objective (OT.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 OT.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 OT.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 OT.LOAD which ensures that the loading of packages is done securely and thus preserves the integrity of applications data. By controlling the access to card management functions such as the installation, update or deletion of applets the objective OT.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 OT.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 OT.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively. Finally, the objective OT.EXT-MEM provides access control for external memory and therefore also contributes to counter this threat.

T.INTEG-JCS-DATA This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (OT.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (OT.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (OT.OPERATE) objective. As the firewall is a software tool automating critical controls, the objective OT.ALARM asks for it to provide clear warning and error messages, so that the appropriate countermeasure can be taken. The objectives OT.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 OT.SCP.RECOVERY and OT.SCP.SUPPORT are intended to support the OT.OPERATE and OT.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter. Finally, the objective OT.EXT-MEM provides access control for external memory and therefore also contributes to counter this threat.

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 (OT.FIREWALL). Uniqueness of subject-identity (OT.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 OT.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 OT.GLOBAL_ARRAYS_CONFID and OT.GLOBAL_ARRAYS_INTEG. The objective OT.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 (OT.SID), the firewall (OT.FIREWALL) and its good working order (OT.OPERATE). The objective OT.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 OT.SCP.RECOVERY and OT.SCP.SUPPORT are intended to support the OT.OPERATE objective of the TOE, so they are indirectly related to the threats that this latter objective contributes to counter.

Unauthorized Execution

T.EXE-CODE.1 Unauthorized execution of a method is prevented by the objective OT.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 OT.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.EXE-CODE-REMOTE The OT.REMOTE security objective contributes to prevent the invocation of a method that is not supposed to be accessible from outside the card.

T.NATIVE This threat is countered by OT.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).

Denial of Service

T.RESOURCES This threat is directly countered by objectives on resource-management (OT.RESOURCES) for runtime purposes and good working order (OT.OPERATE) in a general manner. Consumption of resources during installation and other card management operations are covered, in case of failure, by OT.INSTALL. It should be noticed that, for what relates to CPU usage, the Java Card platform is singlethreaded 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 OT.SCP.RECOVERY and OT.SCP.SUPPORT are intended to support the OT.OPERATE and OT.RESOURCES objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

Card Management

T.DELETION This threat is covered by the OT.DELETION security objective which ensures that both applet and package deletion perform as expected. The objective OT.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 OT.INSTALL which ensures that the installation of an applet performs as expected and the security objectives OT.LOAD which ensures that the loading of a package into the card is safe. The objective OT.CARD-MANAGEMENT controls the access to card management functions and thus contributes to cover this threat.

Services

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

Miscellaneous

T.PHYSICAL Covered by OT.SCP.IC. Physical protections rely on the underlying platform and are therefore an environmental issue.

4.3.1.2 Organisational 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.

4.3.1.3 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.

4.3.2 Security Objectives Rational for Objectives from [10]

4.3.2.1 Threats

T.DF_DATA-MODIFICATION This threat is completely averted by the security objectives OT.DF_DATA-ACCESS and O.DF_TYPE-CONSISTENCY provided by the TOE

T.DF_IMPERSONATE This threat is averted by the security objective OT.DF_AUTHENTICATION. This must be supported by OE.DF_SECURE-VALUES because the authentication is based on keys and the knowlegde of the keys must be limited to the authorized users.

T.DF_CLONING This threat is averted by OT.DF_DATA-ACCESS that prevents the disclosure of sensitive data from the TOE and OT.DF_AUTHENTICATION that limits the access to authorized user only. As already mentioned above, an appropriate key management according to OE.DF_SECURE-VALUES must be ensured.

4.3.2.2 Organisational Security Policys

OSP.DESFire-Emulation The OSP is related to the IC Dedicated Support Software and covers the additional objectives OT.DF_CONFIDENTIALITY, OT.DF_TYPE-CONSISTENCY, and OT.DF_TRANSACTION. Since these objectives require the TOE to implement exactly the same specific security functionality as required by P.DESFire-Emulation, the organizational security policy is covered by the objectives.

4.3.2.3 Assumptions

A.DF_SECURE-VALUES The management of the keys used for the authentication of roles for the DESFire application must be performed outside the TOE. These keys must be loaded in a personalization process and these keys must be protected by the environment. Since OE.DF_SECURE-VALUES requires from the Administrator, Application Manager or the Application User to use secure values for the configuration of the authentication and access control as assumed in A.DF_SECURE-VALUES, the assumption is covered by the objective.

A.DF_TERMINAL-SUPPORT The TOE can only check the integrity of data received from the terminal. For data transferred to the terminal the receiver must verify the integrity of the received data. This is assumed by OE.DF_TERMINAL-SUPPORT, therefore the assumption is covered.

4.3.3 Security Objectives Rationale for Objectives not in [5]

4.3.3.1 Threats

T.OS_OPERATE OT.OPERATE and OT.MF_FW addresses directly the threat T.OS_OPERATE by ensuring the correct continuation of operation of the TOE logical security functions. Security mechanisms have to be implemented to avoid fraudulent usage of the TOE, usage of certain memory regions, or usage of incorrect or unauthorized instructions or commands or sequence of commands. The security mechanisms must be designed to always put the TOE in a known and secure state.

T.SEC_BOX_BORDER OT.SEC_BOX_FW addresses directly the threat T.SEC_BOX_BORDER by ensuring that the native code separated in the Secure Box and the data belonging to this native code is completely sealed off from the Java Card System. Due to the separation the native code in the Secure Box cannot harm the code and data outside the Secure Box

T.RND The objective OT.RND directly covers T.RND. The TOE ensures the cryptographic quality of random number generation. For instance random numbers shall not be predictable and shall have sufficient entropy. Furthermore, the TOE ensures that no information about the produced random numbers is available to an attacker.

4.3.3.2 Organisational Security Policies

OSP.PROCESS-TOE	This organizational security policy is upheld by the security objective for the TOE OT.IDENTIFICATION which ensures that the TOE can be uniquely identified.

4.3.3.3 Assumptions

A.USE_DIAG	This assumption is upheld by the security objective on the operational environment OE.USE_DIAG which guarantees that secure TOE communication protocols are supported and used by the environment.
A.USE_KEYS	This assumption is upheld by the security objective on the operational environment OE.USE_KEYS which guarantees that during the TOE usage, the terminal or system in interaction with the TOE, ensures the protection (integrity and confidentiality) of their own keys by operational means and/or procedures.
A.PROCESS_SEC_IC	This assumption is upheld by the security objective on the operational environment OE. PROCESS_SEC_IC which guarantees protection during composite product manufacturing.

5. Extended Components Definition (ASE_ECD)

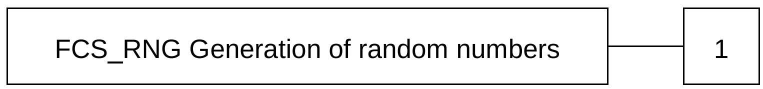
5.1 Definition of Family FCS_RNG

This section has been taken over from the certified (BSI-PP-0035) Smartcard IC Platform Protection profile [6].

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 an random seed to produce a pseudorandom output. A hybrid RNG combines the principles of physical and deterministic RNGs.*

5.2 Definition of the Family FPT_EMSEC

This section has been taken over from the certified (BSI-PP-0017) Protection Profile *Machine Readable travel Document with "ICAO Application", Basic Access Control* [31].

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 [1] part 2.

Family behavior

This family defines requirements to mitigate intelligible emanations.

Component leveling:

FPT_EMSEC TOE emanation	1

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.

5.3 Definition of Family FAU_SAS

This section has been taken over from the certified (BSI-PP-0035) Smartcard IC Platform Protection profile [6].

To define the security functional requirements of the TOE an additional family (FAU_SAS) of the Class FAU (Security Audit) is defined here. This family describes the functional requirements for the storage of audit data. It has a more general approach than FAU_GEN, because it does not necessarily require the data to be generated by the TOE itself and because it does not give specific details of the content of the audit records.

Family behavior

This family defines functional requirements for the storage of audit data.

Component leveling:

FAU_SAS.1	Requires the TOE to provide the possibility to store audit data.
Management:	FAU_SAS.1
	There are no management activities foreseen.
Audit:	FAU_SAS.1
	There are no actions defined to be auditable.
FAU_SAS.1	Audit storage
Hierarchical to:	No other components.
FAU_SAS.1.1	The TSF shall provide [assignment: list of subjects] with the capability to store [assignment: list of audit information] in the [assignment: type of persistent memory].
Dependencies:	No dependencies.

6. Security requirements (ASE_REQ)

This section states the security functional requirements for the TOE. For readability requirements are arranged into groups.

The permitted operations (assignment, iteration, selection and refinement) of the SFRs given in Common Criteria [1] and are printed in **bold**. Completed operations related to the PP are additionally marked within [ ] where assignments are additionally marked with the keyword “assignment”.

Table 16. Requirement Groups

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[5] feature. This group is the union of requirements from the Core (CoreG) and the Logical channels (LCG) groups defined in [6] (cf. Java Card System Protection Profile [5]).
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

5 The PP refers to Java Card Specification 2.2, we use Java Card Specification 3.0.1.

Group	Description
	installed applets from the card, a feature introduced in Java Card specification version 2.2.
Remote Method Invocation (RMIG)	The RMIG contains the security requirements for the remote method invocation feature, which provides a new protocol of communication between the terminal and the applets. This was 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.
Extended Memory (EMG)	The EMG group contains security requirements for the management of external memory

Subjects are active components of the TOE that (essentially) act on the behalf of users. The users of the TOE include people or institutions (like the applet developer, the card issuer, the 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 may be just an alias for the card issuer.

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

Table 17. Subject Descriptions

Subject	Description
S.ADEL	The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([20], §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 §7.1.3.1.
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 §7.1.7.
S.CAD	The CAD represents the actor that requests, by issuing commands to the card, for RMI services. It also plays the role of the 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.

Subject	Description
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	Operand stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.
S.MEMBER	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.
S.ROOTAPP	The root applet behaves like an applet from the user point of view, even though it is part of the OS. It is used in the pre-personalization to configure several parameters of the OS.
S.SBNativeCode	Is the native code library residing in the Secure Box

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

Table 18. Object Descriptions

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.REMOTE_MTHD	A method of a remote interface
O.REMOTE_OBJ	A remote object is an instance of a class that implements one (or more) remote interfaces. A remote interface is one that extends, directly or indirectly, the interface java.rmi.Remote ([19]).
O.RMI_SERVICE	These are instances of the class javacardx.rmi.RMIService. They are the objects that actually process the RMI services.
O.ROR	A remote object reference. It provides information concerning: (i) the identification of a remote object and (ii) the Implementation class of the object or the interfaces implemented by the class of the object. This is the object's information to which the CAD can access.
O.EXT_MEM_INSTANCE	Any External Memory Instance created from the MemoryAccess Interface of the Java Card API [19]
O.SB_Content	The code and data elements of the native code library residing in the Secure Box.
O.NON_SB_Content	Any code and data elements not assigned to the native code

Object	Description
	library residing in the Secure Box
O.SB_SFR	The pool of SFR's assigned to be accessible by native code residing in the Secure Box
O.NON_SB_SFR	All SFR's which are not assigned to the Secure Box. Especially the SFR's used to configure the MMU

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

Table 19. Information Descriptions

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.
I.RORD	Remote object reference descriptors which provide information concerning: (i) the identification of the remote object and (ii) the implementation class of the object or the interfaces implemented by the class of the object. The descriptor is the only object's information to which the CAD can access.

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

Table 20. Security Attribute Descriptions

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.
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 ([20], §4.5.2).
ExportedInfo	Boolean (indicates whether the remote object is exportable or not).
Identifier	The Identifier of a remote object or method is a number that uniquely identifies the remote object or method, respectively.
LC Selection Status	Multiselectable, Non-multiselectable or "None".

Security attribute	Description/Value
LifeTime	CLEAR_ON_DESELECT or PERSISTENT[6].
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.
Remote	An object is Remote if it is an instance of a class that directly or indirectly implements the interface java.rmi.Remote.
Resident Packages	The set of AIDs of the packages already loaded on the card.
Returned References	The set of remote object references that have been sent to the CAD during the applet selection session.
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.
Address space	Accessible memory portion.

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.

Table 21. Operation Descriptions

Operation	Description
OP.ARRAY_ACCESS(O.JAVAOBJECT, field)	Read/Write an array component.
OP.CREATE(Sharing, LifeTime)[7]	Creation of an object (new or makeTransient call).
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.GET_ROR(O.APPLET,...)	Retrieves the initial remote object reference of a RMI based applet. This reference is the seed which

[6] 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.

[7] 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.

Operation	Description
	the CAD client application needs to begin remote method invocations.
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.INVOKE(O.RMI_SERVICE,...)	OP.INVOKE(O.RMI_SERVICE,...) Requests a remote method invocation on the remote object.
OP.JAVA(...)	Any access in the sense of [20], §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)	OP.PUT(S1,S2,I) Transfer a piece of information I from S1 to S2.
OP.RET_RORD(S.JCRE,S.CAD,I.RORD)	OP.RET_RORD(S.JCRE,S.CAD,I.RORD) Send a remote object reference descriptor to the CAD.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see [20], §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.
OP.SB_ACCESS	Any read, write or execution access to a memory area
OP.SB_ACCESS_SFR	Any read/write access to a SFR's

6.1 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 Firewall Policy

6.1.1.1 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,
- OP.ARRAY_ACCESS,
- OP.INSTANCE_FIELD

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.

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.

6.1.1.2 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:

Table 22. Security Attributes

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 ([20], §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 ([20], §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 ([20], §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 ([20], §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:**
  - **The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable",**
  - **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[8] is "Standard".**

FDP_ACF.1.3/FIREWALL

The TSF shall explicitly authorise access of subjects to objects based on the following additional rules:

- **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.**
- **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:

- **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.**

[8] For this operation, there is no accessed object; the “Sharing value” thus refers to the parameter of the operation. This rule simply enforces that shareable transient objects are not allowed. Note: parameters can be seen as security attributes whose value is under the control of the subject. For instance, during the creation of an object, the JavaCardClass attribute’s value is chosen by the creator.

- **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.**

Note: The deletion of applets may render some O.JAVAOBJECT inaccessible, and the Java Card RE may be in charge of this aspect. This can be done, for instance, by ensuring that references to objects belonging to a deleted application are considered as a null reference.

In the case of an array type, fields are components of the array ([13]**,** §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 ([20], §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).

([20] 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.

([20], §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 ([21], §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. ([20], §4).

It should be noted, that the TOE does not support multiple logical channels.

6.1.1.3 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)**.

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.

6.1.1.4 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:

Table 23. Security Attributes

Subject/Object	Security attributes
S.JCVM	**Currently Active Context**
S.LOCAL	**Currently Active Context**
S.MEMBER	**Currently Active Context**
I.DATA	**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 [**assignment: no additional information flow control SFP rules**].

FDP_IFF.1.4/JCVM

The TSF shall explicitly authorise 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**].

Note: The storage of temporary Java Card RE-owned objects references is runtime-enforced ([20], §6.2.8.1-3).

It should be noticed that this policy essentially applies to the execution of bytecode. Native methods[9], 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.

6.1.1.5 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**.

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 [13], §2.5.1.

6.1.1.6 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 (S.JCRE)**.

Note: The modification of the Currently Active Context should be performed in accordance with the rules given in [20], §4 and [21], §3.4.

6.1.1.7 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)**.

Note:The modification of the Currently Active Context should be performed in accordance with the rules given in [20], §4 and [21], §3.4.

6.1.1.8 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**.

6.1.1.9 FMT_MSA.3/FIREWALL Static Attribute Initialisation

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.

[9] For this TOE, there are no native methods.

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 ([20], §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.

6.1.1.10 FMT_MSA.3/JCVM Static Attribute Initialisation

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.

6.1.1.11 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**

6.1.1.12 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.

6.1.2 Application Programming Interface

The following SFRs are related to the Java Card API.

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.

6.1.2.1 FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1.1

The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [**assignment: JCOP RNG**] and specified cryptographic key sizes [**assignment: DES: 112, 168 Bit, RSA: 1976 - 2048 Bit [42], AES: 128, 192, 256 Bit, EC key generation**. **EC: 160, 192, 224, 256, 320 bits with the domain parameters provided in NIST DSS standard FIPS 186-3 [**47] **Appendix D or in Brainpool ECC Standard Curves [37] chapters 3.1 to 3.5.]**] that meet the following: [**assignment: ISO 15946-1-2008 [18]]**

Application note:

(1)The keys can be generated and diversified in accordance with [19]

specification in classes KeyBuilder and KeyPair (at least Session key generation).

(2)RSA key pairs in straightforward format or CRT format are supported. EC_FP is supported but EC_F2M is not supported

(3)This component shall be instantiated according to the version of the Java Card API applying to the security target and the implemented algorithms [19]).

(4)The security functionality is resistant against side channel analysis and similar techniques. It is demonstrated for curves defined by NIST [47] and Brainpool [37] only.

(5) To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

(6)The suggested key length for the RSA algorithm according to BSI TR-02102 [42] is 2000 bits.

6.1.2.2 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: methods: set keys and components of DES, AES, RSA, RSA CRT, secure messaging and EC]** that meets the following: **[assignment: [19], [34]].**

Application note:

- The keys can be accessed as specified in [19] Key class and [34] for proprietary classes.

- This component shall be instantiated according to the version of the Java Card API applying to the security target and the implemented algorithms [19] and [34] for proprietary classes.

6.1.2.3 FCS_CKM.3 Cryptographic Key Access

FCS_CKM.3.1

The TSF shall perform [**assignment: management of DES, AES, RSA, RSA-CRT, and EC-keys**] in accordance with a specified cryptographic key access method [**assignment: methods/commands defined in packages javacard.security of [19] and [34] for proprietary classes**] that meets the following: [**assignment: [19], [34]**].

Application note:

- The keys can be accessed as specified in [19] Key class and [34] for proprietary classes.
- This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms [19] and [34] for proprietary classes.

6.1.2.4 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 by method (e.g. clearKey of [19])**] that meets the following: [**assignment: none**].

Application note:

- The keys are reset as specified in [19] Key class, with the method clearKey(). Any access to a cleared key for ciphering or signing shall throw an exception.
- · This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms [19]).

6.1.2.5 FCS_COP.1 Cryptographic Operation

FCS_COP.1.1/TripleDES

The TSF shall perform **[assignment: data encryption and decryption**] in accordance with a specified cryptographic algorithm **[assignment: Triple-DES in ECB/CBC Mode without padding or with padding method 1 or method 2]** and cryptographic key sizes for 2-key TDES (112 bit) or 3-key TDES (168 bit) that meet the following: **[assignment: ANSI X9.52-1998 [45] (ECB and CBC mode) without Padding, ISO9791-1 padding Method 1, or padding method 2 [27]]**.

Application Notes:

(1) The security functionality is resistant against side channel analysis and similar techniques. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

(2) The CBC mode is to be understood as “outer” CBC mode, i.e. CBC mode as defined in [39] and [45] applied to the block cipher algorithm (either DES or Triple-DES).

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, and 256 Bit**] that meet the following: **[assignment: Advanced Encryption Standard (AES) FIPS Publication 197 [23], NIST Special Publication 800-38A, 2001 [40] (ECB and CBC mode)]**.

Application Notes:

(1) The security functionality is resistant against side channel analysis and similar techniques. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

(2) The CBC mode is to be understood as “outer” CBC mode, i.e. CBC mode as defined in [39] and [45] applied to the block cipher algorithm.

FCS_COP.1.1/ RSACipher

The TSF shall perform [**assignment: data encryption and decryption**] in accordance with a specified cryptographic algorithm [**assignment: RSA encryption/decryption algorithm without or with EME-PKCS1-v1_5 encoding]** and cryptographic key sizes **[assignment: 1976 - 2048 bits**] that meet the following: **[assignment: PKCS #1, v2.1 [24] Section 7.2 (RSAES-PKCS1-v1_5-ENCRYPT, RSAES-PKCS1-v1_5-DECRYPT) and Section 5.1 (RSAEP, RSADP)]**.

Application Notes:

(1) The security functionality is resistant against side channel analysis and similar techniques. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

(2) The input data for the encryption operation is not protected against SCA and fault attacks.

FCS_COP.1.1/ RSASignaturePKCS#1_PSS

The TSF shall perform [**assignment: digital signature generation and verification**] in accordance with a specified cryptographic algorithm **[assignment: RSA signature algorithm with EMSA-PSS encoding and SHA-1, SHA-224 and SHA-256 [41]]** and cryptographic key sizes [**assignment: 1976 - 2048 Bit**] that meet the following: **[assignment: (RSASSA-PSS [24] Section 8.1].**

Application Notes:

(1) The security functionality is resistant against side channel analysis and similar techniques. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

(2) The input data for the encryption operation is not protected against SCA and fault attacks.

FCS_COP.1.1/ RSASignaturePKCS#1

The TSF shall perform [**assignment: digital signature generation and verification**] in accordance with a specified cryptographic algorithm [**assignment: RSA signature algorithm with EMSA-PKCS1-v1_5 encoding and SHA-1 and SHA-256 [41]]** and cryptographic key sizes **[assignment: 1976 - 2048 Bit]** that meet the following: **[assignment: RSASSA-PKCS1-v1.5 [24] Section 8.2].**

Application Notes:

(1) The security functionality is resistant against side channel analysis and similar techniques. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

(2) The input data for the encryption operation is not protected against SCA and fault attacks.

FCS_COP.1.1/ RSASignatureISO9796

The TSF shall perform [**assignment: digital signature generation and verification**] in accordance with a specified cryptographic algorithm [**assignment: RSA SignatureISO9796 with SHA-1, SHA-256 [41]]** and cryptographic key sizes [**assignment: 1976 - 2048 Bit**] that meet the following: [**assignment: ISO/IEC 9796-2:2002 [26]]**.

Application Notes:

(1) The security functionality is resistant against side channel analysis and similar techniques. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).
(2) Message recovery as defined in [26] is not supported.
(3) The input data for the encryption operation is not protected against SCA and fault attacks.

FCS_COP.1.1/ DHKeyExchange

The TSF shall perform [**assignment: Diffie-Hellman key agreement**] in accordance with a specified cryptographic algorithm [**assignment: ECC-DH over GF(p), Diffie–Hellman key exchange** and cryptographic key sizes **[assignment: EC: 160, 192, 224, 256, 320 bits with the domain parameters provided in NIST DSS standard FIPS 186-3 [47] Appendix D or in Brainpool ECC Standard Curves [37] chapters 3.1 to 3.5.** , **1976 – 2048 BIT (PKCS#3)]** that meet the following: **[assignment: for ECC-DH: ISO 11770-3 [25], for PKCS#3 [50]]**.

.

Application Note:

(1) The security functionality is resistant against side channel analysis and similar techniques. It is demonstrated for curves defined by NIST [47] and Brainpool [37] only. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

(2) The supported Diffie-Hellman key exchange algorithm is defined in ISO 11770-3 [25], “Key agreement mechanism 1

FCS_COP.1.1/ DESMAC

The TSF shall perform [**assignment: 8 byte MAC generation and verification**] in accordance with a specified cryptographic algorithm **[assignment: Triple-DES in outer CBC MAC Mode without padding or with padding method 1 or method 2]** and cryptographic key sizes **[assignment: 112, 168 Bit] that meet the following: [assignment: : ISO9797-1 MAC Algorithm 1 without Padding; MAC Algorithm 1 with padding Method 1 or Method 2; MAC Algorithm 3 with padding Method 1 or Method 2 [27]]**.

Application Notes:

(1) The security functionality is resistant against side channel analysis and similar techniques. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

(2) The CBC mode is to be understood as "outer" CBC mode, i.e. CBC mode as defined in [39] and [45] applied to the block cipher algorithm (either DES or Triple-DES). The CBC-MAC mode of operation as defined in ISO 9797-1 [27] MAC Algorithm 1, and also described in Appendix F of [39] is similar to CBC mode, but the output of the CBC-MAC is restricted to the output of the last Triple-DES operation, i.e. only the last block of the ciphertext is returned.
(3) The input of DES CMAC is not protected against fault injection attacks.

FCS_COP.1.1/ AESMAC

The TSF shall perform **[assignment: 16 byte AES-MAC generation and verification**] in accordance with a specified cryptographic algorithm **[assignment: AES-CBC-MAC Mode without Padding]** and cryptographic key sizes **[assignment: 128, 192, 256 Bit]** that meet the following: **[assignment: ISO 9797-1 [27], MAC Algorithm 1 (CBC-MAC mode)].**

Application Notes:

(1) The security functionality is resistant against side channel analysis and similar techniques. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

(2) The CBC mode is to be understood as "outer" CBC mode, i.e. CBC mode as defined in [39] and [45] applied to the block cipher algorithm. The CBC-MAC mode of operation as defined in ISO 9797-1 [27], Algorithm 1, and also described in Appendix F of [39] is similar to CBC mode, but the output of the CBC-MAC is restricted to the output of the last AES operation, i.e. only the last block of the ciphertext is returned.

FCS_COP.1.1/ ECSignature

The TSF shall perform [**assignment: digital signature generation and verification**] in accordance with a specified cryptographic algorithm [**assignment: ECDSA with SHA-1, SHA-224 and SHA-256 [41]]** and cryptographic key sizes [**assignment: EC: 160, 192, 224, 256, 320 bits with the domain parameters provided in NIST DSS standard FIPS 186-3 [47] Appendix D or in Brainpool ECC Standard Curves [37] chapters 3.1 to 3.5.**] that meet the following: [**assignment: ISO 14888-3 [28] and FIPS 186-3 [47] (ECDSA)**].

Application Note:

The security functionality is resistant against side channel analysis and similar techniques. It is demonstrated for curves defined by NIST [47] and Brainpool [37] only. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

FCS_COP.1.1/ ECAdd

The TSF shall perform [**assignment: secure point addition**] in accordance with a specified cryptographic algorithm [**assignment: ECC over GF(p), EC point addition**] and cryptographic key sizes sizes [**assignment: EC: 160, 192, 224, 256, 320 bits with the domain parameters provided in NIST DSS standard FIPS 186-3 [47] Appendix D or in Brainpool ECC Standard Curves [37] chapters 3.1 to 3.5.]** that meet the following: [**assignment: ISO 14888-3 [28]].**

Application Notes:

(1) The input and output values of this function have to be treated as secret values.
(2) The security functionality is resistant against side channel analysis and similar techniques. It is demonstrated for curves defined by NIST [47] and Brainpool [37] only. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

FCS_COP.1.1/ SHA-1

The TSF shall perform [**assignment: secure hash computation**] in accordance with a specified cryptographic algorithm [**assignment: SHA-1**] and cryptographic key sizes [**assignment: none**] that meet the following: [**assignment: FIPS 180-3 [29] Section 6**].

Application note:

The SHA and SHA-2 functions provide limited side channel resistance when the same input is used for a limited number of times. The composite evaluator is advised to consult the ETR for composition when performing composite certifications.

FCS_COP.1.1/ SHA-224

The TSF shall perform [**assignment: secure hash computation**] in accordance with a specified cryptographic algorithm [**assignment: SHA-224**] and cryptographic key sizes [**assignment: none**] that meet the following: [**assignment: FIPS 180-3 [29] Section 6**].

Application note:

The SHA and SHA-2 functions provide limited side channel resistance when the same input is used for a limited number of times. The composite evaluator is advised to consult the ETR for composition when performing composite certifications.

FCS_COP.1.1/ SHA-256

The TSF shall perform [**assignment: secure hash computation**] in accordance with a specified cryptographic algorithm [**assignment: SHA-256**] and cryptographic key sizes [**assignment: none**] that meet the following: [**assignment: FIPS 180-3 [29] Section 6**].

Application note:

The SHA and SHA-2 functions provide limited side channel resistance when the same input is used for a limited number of times. The composite evaluator is advised to consult the ETR for composition when performing composite certifications.

FCS_COP.1.1/ AES_CMAC

The TSF shall perform [**assignment: message authentication and verification**] in accordance with a specified cryptographic algorithm [**assignment: AES - CMAC**] and cryptographic key sizes [**assignment: 128, 192, 256 bit**] that meet the following: [**assignment: Advanced Encryption Standard (AES) FIPS Publication 197 [23], NIST Special Publication 800-38B [32], Section 5 and 6**].

Application notes:

The security functionality is resistant against side channel analysis and similar techniques. To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

FCS_COP.1.1/ TDES_CMAC

The TSF shall perform [**assignment: message authentication and verification**] in accordance with a specified cryptographic algorithm [**assignment: Triple DES-CMAC**] and cryptographic key sizes [**assignment: 112 and 168 bit**] that meet the following: [**assignment: ANSI X9.52-1998 [45] (ECB and CBC mode), NIST Special Publication 800-38B [32], Section 5 and 6**].

Application notes:

(1) The TOE shall provide a subset of cryptographic operations defined in [19] (see javacardx.crypto.Cipher and javacardx.security packages).

(2) This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms [19]).

(3) To fend off attackers with high attack potential an adequate key length must be used (references can be found in national and international documents and standards).

6.1.2.6 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 **de-allocation 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 [20], §5.1.

6.1.2.7 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.

6.1.2.8 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 **de-allocation 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.

6.1.2.9 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 **de-allocation 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 [19].

6.1.2.10 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 **de-allocation 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 [20], §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 ([20], §4.2.

6.1.2.11 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 [20], §7.7, within the bounds of the Commit Capacity ([20], §7.8), and those described in [19]**.

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 [19] (see for instance, PIN-blocking, PIN-checking, update of Transient objects).

6.1.3 Card Security Management

6.1.3.1 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: apply a set of rules to monitor and audit these events and based upon these rules indicate a potential violation of the enforcement of the SFRs]**

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[10] 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(), [19] and ([20], §7.6.2)
- violation of the Firewall or JCVM SFPs,
- unavailability of resources,
- array overflow,
- [**assignment: Card Manager life cycle state (OP_READY, INITIALIZED, SECURED, CARD_LOCKED, TERMINATED) inconsistency audited through the life cycle checks in all administrative operations and the self test mechanism on start-up,**
- **OS Internal life cycle state (FUSED, PROTECTED) inconsistency audited through the life cycle checks in all administrative operations,**
- **Abnormal environmental conditions (frequency, voltage, temperature),**
- **Physical tampering,**
- **EEPROM failure audited through exceptions in the read/write operations and consistency/integrity check,**
- **Corruption of check-summed objects,**
- **Access violation, access to memory not defined as accessible or available].**

Application note:

- **The developer shall provide the exhaustive list of actual potential security vio**lations the TOE reacts to. For instance, other runtime errors related to applet's failure like uncaught exceptions.
- The bytecode verification defines a large set of rules used to detect a "potential security violation". The actual monitoring of these "events" within the TOE only makes sense when the bytecode verification is performed on-card.
- Depending on the context of use and the required security level, there are cases where the card manager and the TOE must work in cooperation to detect and appropriately react in case of potential security violation. This behavior must be described in this component. It shall detail the nature of the feedback information provided to the card manager (like the identity of the offending application) and the conditions under which the feedback will occur (any occurrence of the java.lang.SecurityException exception).
- The "locking of the card session" may not appear in the policy of the card manager. Such measure should only be taken in case of severe violation detection; the same holds for the re-initialization of the Java Card System. Moreover, the locking should occur when "clean" re-initialization seems to be impossible.
- The locking may be implemented at the level of the Java Card System as a denial of service (through some systematic "fatal error" message or return value) that

[10] Applet life cycle states are INSTALLED, SELECTABLE, LOCKED. In addition to these Application Life Cycle States, the Application may define its own Application dependent states.

lasts up to the next "RESET" event, without affecting other components of the card (such as the card manager). Finally, because the installation of applets is a sensitive process, security alertsin this case should also be carefully considered herein.

6.1.3.2 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: D.APP_CODE, D.APP_I_DATA, D.PIN, D.APP_KEYs**].

FDP_SDI.2.2

Upon detection of a data integrity error, the TSF shall [**assignment: maintain a secure state and return an error message**].

Application note:

- Although no such requirement is mandatory in the Java Card specification, at least an exception shall be raised upon integrity errors detection on cryptographic keys, PIN values and their associated security attributes. Even if all the objects cannot be monitored, cryptographic keys and PIN objects shall be 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 shall be monitored (D.APP_I_DATA): applications may 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 could be 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.

6.1.3.3 FPR_UNO.1 Unobservability

FPR_UNO.1.1

The TSF shall ensure that [**assignment: subjects S.Package**] are unable to observe the operation [**assignment: all operations**] on [**assignment: secret keys and PIN codes**] by [**assignment: other subjects S.Package**].

Application note:

Although it is not required in [20] 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 should at least concern secret keys and PIN codes when they exists on the card, as well as the cryptographic operations and comparisons performed on them.

6.1.3.4 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 ([20], §6.2.3) or after a proximity card (PICC) activation sequence ([20]). Behavior of the TOE on power loss and reset is described in [20], §3.6 and §7.1. Behavior of the TOE on RF signal loss is described in [20], §3.6.1.

6.1.3.5 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 [21] specification,
- the API tokens defined in the export files of reference implementation,
- [**assignment: The ISO 7816-6 rules**]
- [**assignment: The EMV 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.

6.1.4 Aid Management

6.1.4.1 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 ([21], §6.5).**

Refinement: "Individual users" stand for applets.

6.1.4.2 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.

6.1.4.3 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: rules defined in FDP_ACF.1.1/FIREWALL, FMT_MSA.2.1/FIREWALL_JCVM and FMT_MSA.3.1/FIREWALL and corresponding application notes**].

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: rules defined in FMT_MSA.1.1/JCRE**].

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".

6.1.4.4 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.

6.1.4.5 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**.

6.1.5 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 PP, loading a package or installing an applet modelled 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).

6.1.5.1 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 ([21], §4.5.2).

6.1.5.2 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.

6.1.5.3 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 [20] §11.1.4**.

Application note:

The TOE may provide additional feedback information to the card manager in case of potential security violations (see FAU_ARP.1).

6.1.5.4 FPT_RCV.3/Installer Automated recovery without undue loss

FPT_RCV.3.1/Installer When automated recovery from **[assignment: a failure during load/installation of a package/applet]** 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: a failure during load/installation of a package/applet]**, 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: 0%]** 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.

6.1.6 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.

6.1.6.1 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.

6.1.6.2 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:

Table 24. Security Attributes

Subject/Object	Security 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, Remote**

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 ([20], §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 ([20], §8.5) O.JAVAOBJECT is remote reachable.**
- **R.JAVA.15 ([20], §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 ([20], §8.5) O.JAVAOBJECT is remote reachable.**
- **R.JAVA.16 ([20], §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 ([20], §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 ([20], §8.5) O.JAVAOBJECT is remote reachable.**

FDP_ACF.1.3/ADEL

The TSF shall explicitly authorise 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.

6.1.6.3 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 [20], §11.3.4.1, §11.3.4.2 and §11.3.4.3.

6.1.6.4 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**.

6.1.6.5 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.

6.1.6.6 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**.

6.1.6.7 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.

6.1.6.8 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 [20], §11.3.4**.

Application note:

- The TOE may provide 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 must comply with Java Card specification ([20], §11.3.4.)

6.1.7 RMIG Security Functional Requirements

This group specifies the policies that control the access to the remote objects and the flow of information that takes place when the RMI service is used. The rules relate mainly to the lifetime of the remote references. Information concerning remote object references can be sent out of the card only if the corresponding remote object has been designated as exportable. Array parameters of remote method invocations must be allocated on the card as global arrays. Therefore, the storage of references to those arrays must be restricted as well. The JCRMI policy embodies both an access control and an information flow control policy.

6.1.7.1 FDP_ACC.2/JCRMI Complete access control

FDP_ACC.2.1/JCRMI

The TSF shall enforce the **JCRMI access control SFP** on **S.CAD, S.JCRE, O.APPLET, O.REMOTE_OBJ, O.REMOTE_MTHD, O.ROR, O.RMI_SERVICE** and all operations among subjects and objects covered by the SFP.

Refinement:

The operations involved in this policy are:

- OP.GET_ROR,
- OP.INVOKE.

FDP_ACC.2.2/JCRMI

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.

6.1.7.2 FDP_ACF.1/JCRMI Security attribute based access control

FDP_ACF.1.1/JCRMI

The TSF shall enforce the **JCRMI access control SFP** to objects based on the following:

Table 25. Security Attributes

Subject/Object	Security attributes
S.JCRE	**Selected Applet Context**
O.REMOTE_OBJ	**Owner, Class, Identifier, ExportedInfo**
O.REMOTE_MTHD	**Identifier**
O.RMI_SERVICE	**Owner, Returned References**

FDP_ACF.1.2/JCRMI

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

- none

FDP_ACF.1.3/JCRMI

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

FDP_ACF.1.4/JCRMI [Editorially Refined] [Editorially Refined NXP]

The TSF shall explicitly deny access of **any subject to O.REMOTE_OBJ and O.REMOTE_MTHD for the purpose of performing a remote method invocation**.

6.1.8 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.

6.1.8.1 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 deallocation after the invocation of the method are described in [19].
- 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.

6.1.8.2 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).

6.1.9 CARG 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.

6.1.9.1 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 **application packages** 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 **application package contained in** 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: at the time when the package is received because no evidence is kept on the card for future verifications]**.

6.1.9.2 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.

6.1.9.3 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:**

1. The keys used by S.BCV, S.CAD, and S.PACKAGE(CM) to secure the communication channel. 2. Authentication retry counter]

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:**

1. S.PACKAGE(CM) should only accept packages sent by S.CAD after S.CAD has been authenticated

2. S.PACKAGE(CM) should only accept packages from S.CAD for which all APDUS have been received and are unmodified and in the correct order].

FDP_IFF.1.3/CM

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

FDP_IFF.1.4/CM

The TSF shall explicitly authorise an information flow based on the following rules: **[assignment: none]**.

FDP_IFF.1.5/CM

The TSF shall explicitly deny an information flow based on the following rules: **[assignment: If the authentication retry counter has reached its maximum number of 66]**.

6.1.9.4 FDP_UIT.1/CM Data exchange integrity

FDP_UIT.1.1/CM

The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to **[receive]** user data in a manner protected from **[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 should be 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.

6.1.9.5 FIA_UID.1/CM Timing of identification

FIA_UID.1.1/CM

The TSF shall allow **[assignment: the following TSF mediated command]** on behalf of the user to be performed before the user is identified.

Table 26. TSF mediated commands for FIA_UID.1

Command	Objects
Get Data	ISD DATA [ISSUER IDENTIFICATION NUMBER], ISD DATA [CARD IMAGE NUMBER], PLATFORM DATA [CARD RECOGNITION DATA], ISD DATA [KEY INFORMATION TEMPLATE], ISD DATA [SCP INFORMATION], PLATFORM DATA [MANUFACTURING]
Select Applet	
Initialize Update	APDU BUFFER
External Authenticate	APDU BUFFER
Identify	

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.

6.1.9.6 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 **[modify], [assignment: create]** the security attributes **[assignment: keys used to secure the communication between S.PACKAGE(CM) and S.CAD]** to **[assignment: S:PACKAGE(CM)]**.

Note: This requirement is no contradiction to FDP_ACF.1/LifeCycle (which allows S.ROOTAPP to manipulate keys) because FMT_MSA.1/CM describes the behaviour starting with the OS Internal Life Cycle State FUSED which is mandatory for phase 7 of the lyfe cycle model.

6.1.9.7 FMT_MSA.3/CM Static attribute initialisation

FMT_MSA.3.1/CM

The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/CM

The TSF shall allow the **[assignment: none]** to specify alternative initial values to override the default values when an object or information is created.

6.1.9.8 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:**

- **modification and creation of the keys used to secure the communication between S.PACKAGE(CM) and S.CAD**

- **modify the behaviour of functions, modify the list of registered applets' AID, modify the card life cycle state attribute**

].

6.1.9.9 FMT_SMR.1/CM Security roles

FMT_SMR.1.1/CM

The TSF shall maintain the roles **[assignment: S.PACKAGE(CM), S.ROOTAPP]**.

FMT_SMR.1.2/CM

The TSF shall be able to associate users with roles.

6.1.9.10 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.

6.1.10 EMG Security Functional Requirements

This group includes requirements for managing the external memory.

6.1.10.1 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.**

6.1.10.2 FDP_ACF.1/EXT_MEM Security attribute based access control

FDP_ACF.1.1/EXT_MEM

The TSF shall enforce the **EXTERNAL MEMORY access control SFP** to objects based on the following: **object O.EXT_MEM_INSTANCE and security attribute 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 authorise 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].**

6.1.10.3 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**.

6.1.10.4 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.

6.1.10.5 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**

6.1.11 Further Functional Requirements not contained in [5]

6.1.12 SCPG Security Functional Requirements

For this evaluation the smart card platform belongs to the TOE and the functional requirements are stated here as functional requirements for the TOE.

6.1.12.1 FPT_FLS.1/SCP Failure with preservation of a Secure State

This assignment operation of the functional requirement has been taken over from the ST of the certified hardware platform P5CD081V1D that is conformant to [6].

FPT_FLS.1.1/SCP

The TSF shall preserve a secure state when the following types of failures occur: **[assignment: exposure to operating conditions which may not be tolerated**

according to the requirement Limited fault tolerance (FRU_FLT.2/SCP) and where therefore a malfunction could occur].

6.1.12.2 FRU_FLT.2/SCP Limited Fault Tolerance

This assignment operation of the functional requirement has been taken over from the ST of the certified hardware platform P5CD081V1D that is conformant to [6].

FRU_FLT.2.1/SCP

The TSF shall ensure the operation of all the TOE capabilities when the following failures occur: [**assignment: exposure to operating conditions which may not be tolerated according to the requirement Failure with preservation of a secure state (FPT_FLS.1/SCP)**].

Refinement: The term “failure” above means “circumstances”. The TOE prevents failures for the “circumstances” defined above.

6.1.12.3 FPT_PHP.3/SCP Resistance to Physical Attack

This functional requirement has been taken over from the ST of the certified hardware platform P5CD081V1D that is conformant to [6].

FPT_PHP.3.1/SCP

The TSF shall resist **[assignment: physical manipulation and physical probing]** to the **[assignment: TSF]** by responding automatically such that the SFRs are always enforced.

Refinement: The TOE will implement appropriate measures to continuously counter physical manipulation and physical probing. Due to the nature of these attacks (especially manipulation) the TOE can by no means detect attacks on all of its elements. Therefore, permanent protection against these attacks is required ensuring that the TSP could not be violated at any time. Hence, “automatic response” means here (i) assuming that there might be an attack at any time and (ii) countermeasures are provided at any time.

6.1.12.4 FDP_ACC.1/SCP Subset Access Control

This functional requirement has been taken over from the ST of the certified hardware platform P5CD081V1D that is conformant to [6].

FDP_ACC.1.1/SCP

The TSF shall enforce the [**assignment: Access Control Policy**] on [**assignment: all code running on the TOE, all memories and all memory operations**].

Application note: The Access Control Policy shall be enforced by implementing a MMU, which maps virtual addresses to physical addresses. The CPU always uses virtual addresses, which are mapped to physical addresses by the MMU. Prior to accessing the respective memory address, the MMU checks if the access is allowed.

6.1.12.5 FDP_ACF.1/SCP Security Attribute based Access Control

This functional requirement has been taken over from the ST of the certified hardware platform P5CD081V1D that is conformant to [6].

FDP_ACF.1.1/SCP

The TSF shall enforce the [**assignment: Access Control Policy**] to objects based on the following: [**assignment: all subjects and objects and the attributes CPU mode,**

the MMU Segment Table, the Special Function Registers to configure the MMU segmentation and the Special Function Registers related to system management].

FDP_ACF.1.2/SCP

The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [***assignment:***

Code executed in the Boot Mode

- **has read and execute access to all code/data in the Test-ROM,**
- **has read, write and execute access to all code/data in the MIFARE-EEPROM**
- **has read and write access to all data in the MIFARE-RAM**

Code executed in the Test Mode

- **has read and execute access to all code/data in the whole ROM,**
- **has read, write and execute access to all code/data in the whole EEPROM**
- **has read and write access to all data in the whole RAM**

Code executed in the MIFARE Mode

- **has read and execute access to all code/data in the Test-ROM,**
- **has read, write and execute access to all code/data in the MIFARE-EEPROM**
- **has read and write access to all data in the MIFARE-RAM**

Code executed in the System Mode

- **has read and execute access to all code/data in the Application-ROM,**
- **has read, write and execute access to all code/data in the Application-EEPROM,**
- **has read and write access to all data in the Application-RAM,**

Code executed in the User Mode

- **has read and/or execute access to code/data in the Application-ROM controlled by the MMU Segment Table used by the MMU,**
- **has read and/or write and/or execute access to code/data in the Application-EEPROM controlled by the MMU Segment Table used by the MMU,**
- **has read and/or write access to data in the Application-RAM controlled by the MMU Segment Table used by the MMU.**]

FDP_ACF.1.3/SCP

The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [**assignment: Code running in MIFARE Mode has read access to 64 bytes in the Application-ROM storing the "Access Condition Matrix". Code running in MIFARE Mode has access to the Application-RAM defined by the Special Function Register MXBASL, MXBASH, MXSZL and MXSZH. Code running in Boot Mode or MIFARE Mode has read access to the Security Row stored in the Application-EEPROM. The FameXE co-processor has read access to the EEPROM and read/write access to the FameXE RAM.**]

FDP_ACF.1.4/SCP

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

6.1.12.6 FMT_MSA.3/SCP Static Attribute Initialization

FMT_MSA.3.1/SCP

The TSF shall enforce the **[assignment: Access Control Policy]** to provide **[selection: restrictive]** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/SCP

The TSF shall allow **[assignment: no subject]** to specify alternative initial values to override the default values when an object or information is created.

Application note: Restrictive means here that the reset values of the Special Function Register regarding the address of the MMU Segment Table are set to zero, which effectively disables any memory segment so that no User Mode code can be executed by the CPU. Furthermore the memory partition cannot be configured at all.

The TOE does not provide objects or information that can be created, since it provides access to memory areas. The definition of objects that are stored in the TOE's memory is subject to the Smartcard Embedded Software.

6.1.13 LifeCycle Security Functional Requirements

This group contains the security requirements for life cycle control mechanism. For this evaluation the life cycle management belongs to the TOE and the functional requirements are stated here as functional requirements for the TOE. Beside the global platform life cycle states defined in [15] Section 5.1. the systems has an OS Internal Life Cycle which defines the following states: no specific state, FUSED and PROTECTED.

6.1.13.1 FDP_ACC.1/LifeCycle Subset Access Control

FDP_ACC.1.1/ LifeCycle

The TSF shall enforce the **[assignment: LIFE CYCLE MANAGEMENT access control SFP]** on **[assignment: subjects: S.ROOTAPP, S.PACKAGE(CM), S.PACKAGE, S.JCRE; objects: D.ADMIN_CONF_DATA, D.PERSO_CONF_DATA, and all operations among subjects and objects covered by the SFP**].

6.1.13.2 FDP_ACF.1/LifeCycle Security Attribute based Access Control

FDP_ACF.1.1/LifeCycle

The TSF shall enforce the **[assignment: LIFE CYCLE MANAGEMENT access control SFP]** to objects based on [**assignment: the security attributes of S.PACKAGE(CM): Card Life Cycle State as defined in [15] Section 5.1: OP_READY, INITIALIZED, SECURED, CARD_LOCKED, TERMINATED, OS Internal Life Cycle States: PROTECTED, FUSED, and the security attributes of S.ROOTAPP: AUTHENTICATED_ADMIN, AUTHENTICATED_TRANSPORT**].

FDP_ACF.1.2/LifeCycle

The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [**assignment:**

1) **S.PACKAGE(CM) is allowed to set the Card Life Cycle OP_READY, INITIALIZED, SECURED, CARD_LOCKED, and TERMINATED.**
2) **S.JCRE is allowed to set the Card Life Cycle to TERMINATED.**

3) **S.ROOTAPP is allowed to set the OS Internal Life Cycle States PROTECTED and FUSED**

4) **S.ROOTAPP is allowed to read and write D.ADMIN_CONF_DATA and D.PERSO_CONF_DATA in the state AUTHENTICATED_ADMIN**

5) **S.ROOTAPP is allowed to read and write D.PERSO_CONF_DATA in the state AUTHENTICATED_TRANSPORT**].

FDP_ACF.1.3/LifeCycle

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

FDP_ACF.1.4/LifeCycle

The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [**assignment:**

6) **If the card life cycle state is TERMINATED, the TOE is blocked, and the access of subjects is no more allowed.**

7) **If the OS Internal Life Cycle is FUSED the TOE blocks any read or write access by S.ROOTAPP]**

6.1.13.3 FMT_MSA.1/LifeCycle Management of Security Attributes

FMT_MSA.1.1/LifeCycle

The TSF shall enforce the [**assignment: LIFE CYCLE MANAGEMENT access control SFP]** to restrict the ability to **[selection: modify**] the security attributes [**assignment: card life cycle state**] to [**assignment: S.PACKAGE(CM)**] and the security attributes [**assignment: OS Internal Life Cycle States**] to [**assignment: S.ROOTAPP**].

.

6.1.13.4 FMT_MSA.3/LifeCycle Static Attribute Initialization

FMT_MSA.3.1/LifeCycle

The TSF shall enforce the [**assignment:LIFE CYCLE MANAGEMENT access control SFP]** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/LifeCycle

The TSF shall allow the [**assignment: no roles**] to specify alternative initial values to override the default values when an object or information is created.

6.1.14 Further Functional Requirements

6.1.14.1 FIA_AFL.1/PIN Basic Authentication Failure Handling

FIA_AFL.1.1/PIN

The TSF shall detect when **[selection: an administrator configurable positive integer within [1 and 127]]** unsuccessful authentication attempts occur related to **[assignment: any user authentication using D.PIN].**

FIA_AFL.1.2/PIN

When the defined number of unsuccessful authentication attempts has been **surpassed**, the TSF shall **[assignment: block the authentication with D.PIN].**

Note: The dependency with FIA_UAU.1 is not applicable. The TOE implements the firewall access control SFP, based on which access to the object implementing FIA_AFL.1/PIN is organized.

.

6.1.14.2 FTP_ITC.1/ LifeCycle Inter-TSF Trusted Channel

FTP_ITC.1.1/LifeCycle

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/ LifeCycle

The TSF shall permit **[assignment: another trusted IT product]** to initiate communication via the trusted channel.

FTP_ITC.1.3/ LifeCycle

The TSF shall initiate communication via the trusted channel for **[assignment: setting the Card Life Cycle State and setting the OS Internal Life Cycle State]**.

6.1.14.3 FAU_SAS.1/SCP Audit Data Storage

This functional requirement has been taken over from the ST of the certified hardware platform P5CD081V1D that is conformant to [6].

FAU_SAS.1.1/SCP

The TSF shall provide [**assignment: test personnel before TOE Delivery**] with the capability to store the [**assignment: Initialisation Data and/or Prepersonalisation Data and/or supplements of the Smartcard Embedded Software**] in the [**assignment: audit records**].

6.1.14.4 FCS_RNG.1 Quality metric for Random Numbers

FCS_RNG.1.1

The TSF shall provide a **deterministic** random number generator that implements:

- Class DRG.3 of **[8].**
- (DRG.3.1) If initialized with a random seed **[selection: [assignment: using the PTRNG of the HW platform conform to class P2 in AIS31 [35]]]**, the internal state of the RNG shall **[selection: have at least 100 bit MIN entropy]**.
- (DRG.3.2) The RNG provides forward secrecy.
- (DRG.3.3) The RNG provides enhanced backward secrecy
-

FCS_RNG.1.2

The TSF shall provide random numbers that meet

- Class DRG.3 of **[8].**
- (DRG.3.4) The RNG initialized with a random seed **[assignment: initialization is initiated at startup when the first APDU is received using the PTRNG of the HW platform conform to class P2 in [35]],** generates output for which **[assignment: 2^{35}]** strings of bit length 128 are mutually different with probability above **[assignment: $1-2^{-37}$]**.

- (DRG.3.5) Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A **[assignment: no additional tests]**.

Application note(1):

(DRG.3.1, DRG.3.4) With perspective to DRNG seeding with P2 and PTG.2 can be considered as equivalent [35].

Application note(2):

The selection of the DRNG class is done via fab-key settings during pre-personalization which allows running the DRNG either as class DRG.3 or class DRG.2 compatible DRNG. The default setting is DRG.2

6.1.14.5 FCS_RNG.1/RNG2 Quality metric for Random Numbers

FCS_RNG.1.1/RNG2

The TSF shall provide a **deterministic** random number generator that implements:

- Class DRG.2 of **[8].**
- (DRG.2.1) If initialized with a random seed **[selection: [assignment: using the PTRNG of the HW platform conform to class P2 in AIS31 [35]]**, the internal state of the RNG shall **[selection: have at least 100 bit MIN entropy]**.
- (DRG.2.2) The RNG provides forward secrecy.
- (DRG.2.3) The RNG provides backward secrecy.

FCS_RNG.1.2/RNG2

The TSF shall provide random numbers that meet

- Class DRG.2 of **[8].**
- (DRG.2.4) The RNG initialized with a random seed **[assignment initialization is initiated at startup when the first APDU is received using the PTRNG of the HW platform conform to class P2 in [35]]**, generates output for which **[assignment: 2^{35}]** strings of bit length 128 are mutually different with probability above **[assignment: $1\text{-}2^{-37}$]**.
- (DRG.2.5) Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A **[assignment: no additional tests]**.

Application note(1):

(DRG.2.1, DRG.2.4) With perspective to DRNG seeding with P2 and PTG.2 can be considered as equivalent [35].

Application note(2):

The selection of the DRNG class is done via fab-key settings during pre-personalization which allows running the DRNG either as class DRG.3 or class DRG.2 compatible DRNG. The default setting is DRG.2.

6.1.14.6 FPT_EMSEC.1 TOE Emanation

FPT_EMSEC.1.1

The TOE shall not emit **[assignment: variations in power consumption or timing during command execution]** in excess of **[assignment: non-useful information]** 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 unauthorized]** *users* are unable to use the following interface **[assignment: electrical contacts]** to gain access to **[assignment: TSF data: D.JCS_KEYs and D.CRYPTO]** and **[assignment: User data: D.PIN, D.APP_KEYs].**

6.1.15 Functional Requirements for the Secure Box

This group contains the functional requirements for the Secure Box which is part of the TOE.

6.1.15.1 FDP_ACC.2/SecureBox Complete Access Control

FDP_ACC.2.1/SecureBox

The TSF shall enforce the **[assignment: Secure Box access control SFP]** on **[assignment: S.SBNativeCode, O.SB_Content, O.NON_SB_Content, O.SB_SFR, O.NON_SB_SFR]** and all operations among subjects and objects covered by the SFP.

Refinement:

The operations involved in the policy are:

- OP.SB_ACCESS
- OP.SB_ACCESS_SFR

FDP_ACC.2.2/SecureBox

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.

6.1.15.2 FDP_ACF.1/SecureBox Security Attribute based Access Control

FDP_ACF.1.1/ SecureBox

The TSF shall enforce the **[assignment: Secure Box access control SFP]** to all objects based on the following: **[assignment: S.SBNativeCode, O.SB_Content, O.NON_SB_Content, O.SB_SFR, O.NON_SB_SFR and the attributes CPU mode, the MMU Segment Table, the Special Function Registers to configure the MMU segmentation and the Special Function Registers related to system management.]**

FDP_ACF.1.2/SecureBox

The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **[assignment:**

- **Code assigned to S.SBNativeCode shall only be executed in User Mode**
- **Code assigned to S.SBNativeCode shall only be able to perform OP.SB_ACCESS to O.SB_CONTENT . The ROM, EEPROM, and RAM which belongs to O.SB_CONTENT is controlled by the MMU Segment Table used by the Memory Management Unit**
- **Code assigned to S.SBNativeCode is able to perform OP.SB_ACCESS_SFR to O.SB_SFR. O.SB_SFR is defined by the access rights defined in the**

respective Memory Segment (O.SB_CONTENT) in the MMU Segment Table from which the code is actually executed.]

FDP_ACF.1.3/SecureBox

The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **[assignment: none]**

FDP_ACF.1.4/SecureBox

The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[assignment:**

- **For S.SBNative Code it shall not be possible to perform OP.SB_ACCESS to O.NON_SB_CONTENT**
- **For S.SBNative Code it shall not be possible to perform OP.SB_ACCESS_SFR to O.NON_SB_SFR]**

6.1.15.3 FMT_MSA.3/SecureBox Static attribute initialisation

FMT_MSA.3.1/SecureBox

The TSF shall enforce the **[assignment: Secure Box access control SFP]** to provide **[selection: restrictive]** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/SecureBox

The TSF shall allow **[assignment: JCRE]** to specify alternative initial values to override the default values when an object or information is created.

Application note: During the prepersonalisation of the TOE the initial restrictive values for the security attributes can be overridden by the JCRE

Application note: The dependency to FMT_SMR.1 is fulfilled by Section 6.1.1.12

6.1.15.4 FMT_MSA.1/SecureBox Management of security attributes

FMT_MSA.1.1/SecureBox

The TSF shall enforce the **[assignment: Secure Box access control SFP]** to restrict the ability **to [selection: modify]** the security attributes **[assignment: CPU Mode and, the MMU Segment Table]** to **[assignment: JCRE].**

Application note: The dependency to FMT_SMR.1 is fulfilled by Section 6.1.1.12

6.1.15.5 FMT_SMF.1/SecureBox Specification of Management Functions

FMT_SMF.1.1/SecureBox

The TSF shall be capable of performing the following management functions: **[assignment:**

- **Switch the CPU Mode**
- **Change the values in the MMU Segment Table to assign RAM to the Secure Box**
- **Change the values in the MMU Segment Table to assign EEPROM to the Secure Box]**

6.1.16 MIFARE DESFire Emulation Functional Requirements from [10]

Table 27 lists all SFR's of the MIFARE DESFire Emulation and their corresponding names in [10]. All these SFR's as well as their dependencies are fulfilled by the underlying hardware thus their details can be found in [10].

Table 27. Security functional requirements of the MIFARE DESFire Emulation and the corresponding SFR in [10]

SFR	Correspondent in [10]
FMT_SMR.1[DESFire]	FMT_SMR.1[DESFire]
FDP_ACC.1[DESFire]	FDP_ACC.1[DESFire]
FDP_ACF.1[DESFire]	FDP_ACF.1[DESFire]
FMT_MSA.3[DESFire]	FMT_MSA.3[DESFire]
FMT_MSA.1[DESFire]	FMT_MSA.1[DESFire]
FMT_SMF.1[DESFire]	FMT_SMF.1[DESFire]
FDP_ITC.2[DESFire]	FDP_ITC.2
FCS_CKM.4[DESFire]	FCS_CKM.4
FMT_MTD.1[DESFire]	FMT_MTD.1
FCS_COP.1[DESFire_HW_DES]	FCS_COP.1[HW_DES]
FCS_COP.1[DESFire_HW_AES]	FCS_COP.1[HW_AES]
FIA_UID.2[DESFire]	FIA_UID.2
FIA_UAU.2[DESFire]	FIA_UAU.2
FIA_UAU.5[DESFire]	FIA_UAU.5
FTP_TRP.1[DESFire]	FTP_TRP.1
FPT_RPL.1[DESFire]	FPT_RPL.1
FPT_TDC.1[DESFire]	FPT_TDC.1
FDP_ROL.1[DESFire]	FDP_ROL.1

6.2 Security Assurance Requirements

The assurance requirements of this evaluation are EAL4 augmented by ALC_DVS.2, ASE_TSS.2 and AVA_VAN.5.

In the following the requirements of EAL4 augmented by ALC_DVS.2, ASE_TSS.2 and AVA_VAN.5 are described.

The assurance requirements ensure, among others, the security of the TOE during its development and production. We present here some application notes on the assurance requirements included in the EAL of the ST.

- **ADV_FSP.4** Complete functional specification

- **ADV_ARC.1** Security architecture description
- **ADV_TDS.3** Basic modular design
- **AGD_OPE.1** Operational user guidance
  These SARs ensure proper installation and configuration: the TOE will be correctly configured and the TSFs will be put in good working order. The administrator is the card issuer, the platform developer, the card embedder or any actor who participates in the fabrication of the TOE once its design and development is complete (its source code is available and released by the TOE designer). The users are applet developers, the card manager developers, and possibly the final user of the TOE.

The applet and API packages programmers should have a complete understanding of the concepts defined in [20] and [21]. They must delegate key management, PIN management and cryptographic operations to dedicated APIs. They should carefully consider the effect of any possible exception or specific event and take appropriate measures (such as catch the exception, abort the current transaction, and so on.). They must comply with all the recommendations given in the platform programming guide as well. Failure to do so may jeopardize parts of (or even the whole) applet and its confidential data.

This guidance also includes the fact that sharing object(s) or data between applets (through shareable interface mechanism, for instance) must include some kind of authentication of the involved parties, even when no sensitive information seems at stake (so-called “defensive development”).

- **AGD_PRE.1** Preparative procedures

This SAR ensures the integrity of the TOE and its documentation during the transfer of the TOE between all the actors appearing in the first two stages. Procedures shall ensure protection of TOE material/information under delivery and storage that corrective actions are taken in case of improper operation in the delivery process and storage and that people dealing with the procedure for delivery have the required skills.

- **ALC_CMC.4** Production support, acceptance procedures and automation
- **ALC_CMS.4** Problem tracking CM coverage.
- **ALC_DEL.1** Delivery procedures
- **ALC_LCD.1** Developer defined life-cycle model
- **ALC_TAT.1** Compliance with implementation standards

It is assumed that security procedures are used during all manufacturing and test operations through the production phase to maintain confidentiality and integrity of the TOE and of its manufacturing and test data (to prevent any possible copy, modification, retention, theft or unauthorized use).

- **ATE_COV.2** Analysis of coverage
- **ATE_DPT.1** Testing: modular design
- **ATE_FUN.1** Functional testing
- **ATE_IND.2** Independent testing - sample

The purpose of these SARs is to ensure whether the TOE behaves as specified in the design documentation and in accordance with the TOE security functional requirements. This is accomplished by determining that the developer has tested the security functions against its functional specification and high level design, gaining confidence in those tests results by performing a sample of the developer's tests, and by independently testing a subset of the security functions.

- **ASE_CCL.1** Conformance claims
- **ASE_ECD.1** Extended components definition
- **ASE_INT.1** ST introduction
- **ASE_OBJ.2** Security objectives
- **ASE_REQ.2** Derived security requirements
- **ASE_SPD.1** Security problem definition
- **ASE_TSS.1** TOE summary specification
  These requirements are covered by this document.

Augmentation of level EAL4 results from the selection of the following three SARs:

- **ALC_DVS.2** Sufficiency of security measures

EAL4 requires for the development security the assurance component ALC_DVS.1. This dictates a documentation and check of the security measures in the development environment. The component **ALC_DVS.2** requires additionally a justification, that the measures provide the necessary level of protection.

- **ASE_TSS.2** TOE summary specification with architectural design summary

EAL4 requires for the development security the assurance component **ASE_TSS.1**. This ensures, that The TOE summary specification describes how the TOE meets each SFR. The component **ASE_TSS.2** requires additionally that the TOE summary specification describes how the TOE protects itself against interference and logical tampering and how the TOE protects itself against bypass.

- **AVA_VAN.5** Advanced methodical vulnerability analysis

EAL4 requires for the vulnerability assessment the assurance component **AVA_VAN.3**. Its aim is to determine whether the TOE, in its intended environment, has vulnerabilities exploitable by attackers processing moderate attack potential. In order to provide the necessary level of protection, EAL4 is augmented with the component **AVA_VAN.5**, which requires that the TOE is resistant against attackers processing high attack potential.

6.3 Security Requirements Rationale

This section proves that the given security requirements (TOE and environment) cover the security objectives described in Section 4.

6.3.1 Security Functional Requirements Rationale for SFRs tables

All security objectives of the TOE are met by the security functional requirements. At least one security objective exists for each security functional requirement.

Table 28. Assignment: Security Objectives for the TOE – Security Requirements 1.

	FDP_ACC.2/FIREWALL	FDP_ACF.1/FIREWALL	FDP_IFC.1/JCVM	FDP_IFF.1/JCVM	FDP_RIP.1/OBJECTS	FMT_MSA.1/JCRE	FMT_MSA.1/JCVM	FMT_MSA.2/FIREWALL_JCVM	FMT_MSA.3/FIREWALL	FMT_MSA.3/JCVM	FMT_SMF.1	FMT_SMR.1	FCS_CKM.1	FCS_CKM.2	FCS_CKM.3	FCS_CKM.4	FCS_COP.1	FDP_RIP.1/ABORT	FDP_RIP.1/APDU	FDP_RIP.1/bArray	FDP_RIP.1/KEYS	FDP_RIP.1/TRANSIENT	FDP_ROL.1/FIREWALL	FAU_ARP.1	FDP_SDI.2
OT.SID						x	x		x	x															
OT.FIREWALL	x	x	x	x		x	x	x	x	x	x	x													
OT.GLOBAL_ARRAYS_CONFID			x	x	x													x	x	x	x	x			
OT.GLOBAL_ARRAYS_INTEG			x	x																					
OT.NATIVE		x																							
OT.OPERATE	x	x																					x	x	
OT.REALLOCATION					x													x	x	x	x	x			
OT.RESOURCES											x	x											x	x	
OT.ALARM																								x	
OT.CIPHER													x	x	x	x	x								
OT.KEY-MNGT					x								x	x	x	x	x	x	x	x	x	x			x
OT.PIN-MNGT	x	x			x													x	x	x	x	x	x		x
OT.REMOTE																									
OT.TRANSACTION					x													x	x	x	x	x	x		
OT.OBJ-DELETION																									
OT.DELETION																									
OT.LOAD																									
OT.INSTALL																									
OT.SCP.IC																								x	
OT.SCP.RECOVERY																									
OT.SCP.SUPPORT													x			x	x						x		
OT.EXT-MEM																									
OT.MF_FW																									
OT.CARD-MANAGEMENT																									

	FDP_ACC.2/FIREWALL	FDP_ACF.1/FIREWALL	FDP_IFC.1/JCVM	FDP_IFF.1/JCVM	FDP_RIP.1/OBJECTS	FMT_MSA.1/JCRE	FMT_MSA.1/JCVM	FMT_MSA.2/FIREWALL_JCVM	FMT_MSA.3/FIREWALL	FMT_MSA.3/JCVM	FMT_SMF.1	FMT_SMR.1	FCS_CKM.1	FCS_CKM.2	FCS_CKM.3	FCS_CKM.4	FCS_COP.1	FDP_RIP.1/ABORT	FDP_RIP.1/APDU	FDP_RIP.1/bArray	FDP_RIP.1/KEYS	FDP_RIP.1/TRANSIENT	FDP_ROL.1/FIREWALL	FAU_ARP.1	FDP_SDI.2
OT.IDENTIFICATION																									
OT.RND																									
OT.SEC_BOX_FW																									

Table 29. Assignment: Security Objectives for the TOE – Security Requirements 2.

	FPR_UNO.1	FPT_FLS.1	FPT_TDC.1	FIA_ATD.1/AID	FIA_UID.2/AID	FIA_USB.1/AID	FMT_MTD.1/JCRE	FMT_MTD.3/JCRE	FDP_ITC.2/Installer	FMT_SMR.1/Installer	FPT_FLS.1/Installer	FPT_RCV.3/Installer	FDP_ACC.2/ADEL	FDP_ACF.1/ADEL	FDP_RIP.1/ADEL	FMT_MSA.1/ADEL	FMT_MSA.3/ADEL	FMT_SMF.1/ADEL	FMT_SMR.1/ADEL	FPT_FLS.1/ADEL	FDP_ACC.2/JCRMI	FDP_ACF.1/JCRMI
OT.SID				x	x	x	x	x	x							x	x	x				
OT.FIREWALL							x	x	x	x						x	x	x	x		x	x
OT.GLOBAL_ARRAYS_CONFID															x							
OT.GLOBAL_ARRAYS_INTEG																						
OT.NATIVE																						
OT.OPERATE		x	x	x		x			x		x	x								x		
OT.REALLOCATION															x							
OT.RESOURCES		x					x	x		x	x	x						x	x	x		
OT.ALARM		x									x									x		
OT.CIPHER	x																					
OT.KEY-MNGT	x														x							
OT.PIN-MNGT	x														x							

	FPR_UNO.1	FPT_FLS.1	FPT_TDC.1	FIA_ATD.1/AID	FIA_UID.2/AID	FIA_USB.1/AID	FMT_MTD.1/JCRE	FMT_MTD.3/JCRE	FDP_ITC.2/Installer	FMT_SMR.1/Installer	FPT_FLS.1/Installer	FPT_RCV.3/Installer	FDP_ACC.2/ADEL	FDP_ACF.1/ADEL	FDP_RIP.1/ADEL	FMT_MSA.1/ADEL	FMT_MSA.3/ADEL	FMT_SMF.1/ADEL	FMT_SMR.1/ADEL	FPT_FLS.1/ADEL	FDP_ACC.2/JCRMI	FDP_ACF.1/JCRMI
OT.REMOTE																					x	x
OT.TRANSACTION															x							
OT.OBJ-DELETION																						
OT.DELETION												x	x	x	x	x	x		x	x		
OT.LOAD																						
OT.INSTALL									x		x	x										
OT.SCP.IC																						
OT.SCP.RECOVERY																						
OT.SCP.SUPPORT																						
OT.EXT-MEM																						
OT.MF_FW																						
OT.CARD-MANAGEMENT																						
OT.IDENTIFICATION																						
OT.RND																						
OT.SEC_BOX_FW																						

Table 30. Assignment: Security Objectives for the TOE – Security Requirements 3.

	FDP_RIP.1/ODEL	FPT_FLS.1/ODEL	FCO_NRO.2/CM	FDP_IFC.2/CM	FDP_IFF.1/CM	FDP_UIT.1/CM	FIA_UID.1/CM	FMT_MSA.1/CM	FMT_MSA.3/CM	FMT_SMF.1/CM	FMT_SMR.1/CM	FTP_ITC.1/CM	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	FPT_FLS.1/SCP	FRU_FLT.2/SCP	FPT_PHP.3/SCP	FDP_ACC.1/SCP	FDP_ACF.1/SCP
OT.SID								x	x	x					x	x	x					
OT.FIREWALL								x	x	x	x				x	x	x					

	FDP_RIP.1/ODEL	FPT_FLS.1/ODEL	FCO_NRO.2/CM	FDP_IFC.2/CM	FDP_IFF.1/CM	FDP_UIT.1/CM	FIA_UID.1/CM	FMT_MSA.1/CM	FMT_MSA.3/CM	FMT_SMF.1/CM	FMT_SMR.1/CM	FTP_ITC.1/CM	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	FPT_FLS.1/SCP	FRU_FLT.2/SCP	FPT_PHP.3/SCP	FDP_ACC.1/SCP	FDP_ACF.1/SCP
OT.GLOBAL_ARRAYS_CONFID	x																					
OT.GLOBAL_ARRAYS_INTEG																						
OT.NATIVE																						
OT.OPERATE		x																				
OT.REALLOCATION	x																					
OT.RESOURCES		x								x	x						x					
OT.ALARM		x																				
OT.CIPHER																			x			
OT.KEY-MNGT	x																					
OT.PIN-MNGT	x																					
OT.REMOTE																						
OT.TRANSACTION	x																					
OT.OBJ-DELETION	x	x																				
OT.DELETION																						
OT.LOAD			x	x	x	x	x					x										
OT.INSTALL																						
OT.SCP.IC																		x	x	x		
OT.SCP.RECOVERY																			x			
OT.SCP.SUPPORT																						
OT.EXT-MEM													x	x			x					
OT.MF_FW																					x	x
OT.CARD-MANAGEMENT																						
OT.IDENTIFICATION																						
OT.RND																						
OT.SEC_BOX_FW																						

Table 31. Assignment: Security Objectives for the TOE – Security Requirements 4.

	FMT_MSA.3/SCP	FDP_ACC.1/LifeCycle	FDP_ACF.1/LifeCycle	FMT_MSA.1/LifeCycle	FMT_MSA.3/LifeCycle	FIA_AFL.1/PIN	FTP_ITC.1/LifeCycle	FAU_SAS.1/SCP	FCS_RNG.1/RNG2	FCS_RNG.1	FPT_EMSEC.1	FDP_ACC.2/SecureBox	FDP_ACF.1/SecureBox	FMT_MSA.3/SecureBox	FMT_MSA.1/SecureBox	FMT_SMF.1/SecureBox
OT.SID																
OT.FIREWALL																
OT.GLOBAL_ARRAYS_CONFID																
OT.GLOBAL_ARRAYS_INTEG																
OT.NATIVE																
OT.OPERATE						x										
OT.REALLOCATION																
OT.RESOURCES																
OT.ALARM																
OT.CIPHER																
OT.KEY-MNGT																
OT.PIN-MNGT																
OT.REMOTE																
OT.TRANSACTION																
OT.OBJ-DELETION																
OT.DELETION																
OT.LOAD																
OT.INSTALL																
OT.SCP.IC											x					
OT.SCP.RECOVERY																
OT.SCP.SUPPORT																
OT.EXT-MEM																
OT.MF_FW	x															
OT.CARD-MANAGEMENT		x	x	x	x		x									
OT.IDENTIFICATION								x								

	FMT_MSA.3/SCP	FDP_ACC.1/LifeCycle	FDP_ACF.1/LifeCycle	FMT_MSA.1/LifeCycle	FMT_MSA.3/LifeCycle	FIA_AFL.1/PIN	FTP_ITC.1/LifeCycle	FAU_SAS.1/SCP	FCS_RNG.1/RNG2	FCS_RNG.1	FPT_EMSEC.1	FDP_ACC.2/SecureBox	FDP_ACF.1/SecureBox	FMT_MSA.3/SecureBox	FMT_MSA.1/SecureBox	FMT_SMF.1/SecureBox
OT.RND									x	x						
OT.SEC_BOX_FW												x	x	x	x	x

6.3.2 Security Functional Requirements Rationale from [5]

The following chapters have been taken from [5] without modifications.

6.3.2.1 Security Objectives for the TOE

Identification

OT.SID (Refined) 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/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. Lastly, installation procedures ensure protection against forgery (the AID of an applet is under the control of the TSFs) or re-use of identities (FIA_UID.2/AID, FIA_USB.1/AID).

Execution

OT.FIREWALL (Refined) 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), the JCRMI access control policy (FDP_ACC.2/JCRMI, FDP_ACF.1/JCRMI) 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.

OT.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).

OT.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.

OT.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.

OT.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). Furthermore authentication is protected by FIA_AFL.1/PIN. Almost every objective and/or functional requirement indirectly contributes to this one too.

OT.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.

OT.RESOURCES (Refined) 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).

Services

OT.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.

OT.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. It is supported by FRU_FLT.2/SCP by preserving a secure state in case of operating conditions which may not be tolerated.

OT.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.

OT.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.

OT.REMOTE (Refined) The access to the TOE's internal data and the flow of information from the card to the CAD required by the JCRMI service is under control of the JCRMI access control policy (FDP_ACC.2/JCRMI, FDP_ACF.1/JCRMI).

OT.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).

Object Deletion

OT.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.

Applet Management

OT.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.

OT.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).

OT.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).

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)

6.3.3 Security Functional Requirements Rationale not from [5]

OT.SCP.RECOVERY This objective is met by the component FRU_FLT.2/SCP

OT.SCP.SUPPORT This objective is met by the components FDP_ROL.1/FIREWALL, FCS_COP.1, FCS_CKM.1, and FCS_CKM.4.

OT.SCP.IC This objective is met by the components FAU_ARP.1, FPT_FLS.1/SCP, FRU_FLT.2/SCP, FPT_PHP.3, and FPT_EMSEC.1.

OT.CARD-MANAGEMENT This objective shall control the access to the card and implement the card issuers policy and is met by the components FDP_ACC.1/LifeCycle, FDP_ACF.1/LifeCycle, FMT_MSA.1/LifeCycle, FMT_MSA.3/LifeCycle, and FTP_ITC.1/LifeCycle.

OT.IDENTIFICATION Obviously the operations for FAU_SAS.1/SCP are chosen in a way that they require the TOE to provide the functionality needed for OT.IDENTIFICATION. The Initialisation Data (or parts of them) are used for TOE identification.

OT.RND OT.RND requires random numbers of a good cryptographic quality. FCS_RNG.1 requires the TOE to provide random numbers of good quality by specifying class DRG.3 or DRG.2 of AIS 20, thus fulfilling OT.RND.

It was chosen to define FCS_RNG.1 explicitly, because Part 2 of the Common Criteria does not contain generic security functional requirements for Random Number generation. (Note that there are security functional requirements in Part 2 of the Common Criteria, which refer to random numbers. However, they define requirements only for the authentication context, which is only one of the possible applications of random numbers.)

OT.MF_FW The access control mechanisms described by OT.MF_FW are directly addressed by the SFP defined by the security functional requirements FDP_ACC.1/SCP, FDP_ACF.1, and FMT_MSA.3/SCP.

OT.SEC_BOX_FW The access control mechanisms described by OT.SEC_BOX_FW are directly addressed by the SFP defined by the security functional requirements FDP_ACC.2/SecureBox, FDP_ACF.1/SecureBox, FMT_MSA.3/SecureBox, FMT_MSA.1/SecureBox, and , FMT_SMF.1/SecureBox.

6.4 SFRs Dependencies

Table 32. SFR dependencies and their fullfilment

SFR	Dep.	Met?
FDP_ITC.2/Installer	[FDP_ACC.1 or FDP_IFC.1] FPT_TDC.1 [FPT_ITC.1 or FTP_TRP.1]	Yes, FDP_IFC.2/CM, FTP_ITC.1/CM, FPT_TDC.1
FMT_SMR.1/Installer	(FIA_UID.1)	No, rationale in Section 0
FPT_FLS.1/Installer	No dependencies	
FPT_RCV.3/Installer	AGD_OPE.1	Yes, AGD_OPE.1

SFR	Dep.	Met?
FDP_ACC.2/ADEL	FDP_ACF.1	Yes, FDP_ACF.1/ADEL
FDP_ACF.1/ADEL	FDP_ACC.1 FMT_MSA.3	Yes, 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] FMT_SMF.1 FMT_SMR.1	Yes, FDP_ACC.2/ADEL, FMT_SMF.1/ADEL, FMT_SMR.1/ADEL
FMT_MSA.3/ADEL	FMT_MSA.1 FMT_SMR.1	Yes, FMT_MSA.1/ADEL, FMT_SMR.1/ADEL
FMT_SMF.1/ADEL	No dependencies	
FMT_SMR.1/ADEL	FIA_UID.1	No, rationale in Section 0
FPT_FLS.1/ADEL	No dependencies	
FDP_ACC.2/JCRMI	FDP_ACF.1	Yes, FDP_ACF.1/JCRMI
FDP_ACF.1/JCRMI	FDP_ACC.1 FMT_MSA.3	No not fully , rationale in Section 0 FDP_ACC.2/JCRMI,
FDP_RIP.1/ODEL	No dependencies	
FPT_FLS.1/ODEL	No dependencies	
FCO_NRO.2/CM	FIA_UID.1	Yes, FIA_UID.1/CM
FDP_IFC.2/CM	FDP_IFF.1	Yes, FDP_IFF.1/CM
FDP_IFF.1/CM	FDP_IFC.1 FMT_MSA.3	Yes, FDP_IFC.1/CM FMT_MSA.3/CM
FDP_UIT.1/CM	[FDO_ACC.1 or FDP_IFC.1] [FTP_ITC1 or FTP_TRP.1]	Yes, 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)	Yes, FDP_IFC.2/CM, FMT_SMF.1/CM, FMT_SMR.1/CM
FMT_MSA.3/CM	FMT_MSA.1 FMT_SMR.1	Yes, FMT_MSA.1/CM, FMT_SMR.1/CM
FMT_SMF.1/CM	No dependencies	
FMT_SMR.1/CM	FIA_UID.1	Yes, FIA_UID.1/CM
FTP_ITC.1/CM	No dependencies	
FDP_ACC.2/FIREWALL	FDP_ACF.1	Yes, FDP_ACF.1/FIREWALL
FDP_ACF.1/FIREWALL	FDP_ACC.1 FMT_MSA.3	Yes, FDP_ACC.2/FIREWALL, FMT_MSA.3/FIREWALL

SFR	Dep.	Met?
FDP_IFF.1/JCVM	FDP_IFC.1 FMT_MSA.3	Yes, FDP_IFC.1/JCVM FMT_MSA.3/JCVM
FDP_IFC.1/JCVM	FDP_IFC.1 FMT_MSA.3	Yes, 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] FMT_SMF.1 FMT_SMR.1	Not fully, rationale in Section 0 FDP_ACC.2/FIREWALL, FMT_SMR.1
FMT_MSA.1/JCVM	[FDP_ACC.1 or FDP_IFC.1] FMT_SMF.1 FMT_SMR.1	Yes, 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] FMT_MSA.1 FMT_SMR.1	Yes, 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 FMT_SMR.1	Yes, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1
FMT_MSA.3/JCVM	FMT_MSA.1 FMT_SMR.1	Yes, FMT_MSA.1/JCVM, FMT_SMR.1
FMT_SMF.1	No dependencies	
FMT_SMR.1	FIA_UID.1	Yes, FIA_UID.2/AID
FCS_CKM.1	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	Yes, FCS_CKM.2, FCS_CKM.4
FCS_CKM.2	[FDP_ITC.1, or FDP_ITC.2, or FCS_CKM.1] FCS_CKM.4	Yes, FCS_CKM.1, FCS_CKM.4
FCS_CKM.3	[FDP_ITC.1, or FDP_ITC.2, or FCS_CKM.1] FCS_CKM.4	Yes, FCS_CKM.1, FCS_CKM.4
FCS_CKM.4 ,	[FDP_ITC.1, or FDP_ITC.2, or FCS_CKM.1] FCS_CKM.4	Yes, FCS_CKM.1,
FCS_COP.1.1/AES	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/TripleDES	[FDP_ITC.1 or	Yes,

SFR	Dep.	Met?
	FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/RSACiper	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/RSASingature ISO9796	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/RSASingaturePKCS#1	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/ RSASingaturePKCS#1_PSS	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/ ECSingature	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/ ECAdd	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/ DHKeyExchange	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/ SHA-1	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/ SHA-224	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/ SHA-256	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/AES_CMAC	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FCS_COP.1.1/TDES_CMAC	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1], FCS_CKM.4	Yes, FCS_CKM.1 and FCS_CKM.4
FDP_RIP.1/ABORT	No dependencies	
FDP_RIP.1/APDU	No dependencies	

SFR	Dep.	Met?
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]	Yes, FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM
FDP_ACC.1/EXT_MEM	FDP_ACF.1	FDP_ACF.1/EXT_MEM
FDP_ACF.1/EXT_MEM	FDP_ACC.1 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] FMT_SMF.1 FMT_SMR.1	FDP_ACC.1/EXT_MEM, FMT_SMF.1/EXT_MEM, FMT_SMR.1
FMT_MSA.3/EXT_MEM	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1/EXT_MEM, FMT_SMR.1
FMT_SMF.1/EXT_MEM	No dependencies	
FAU_ARP.1	FAU_SAA.1	No, rationale in Section 0
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	Yes, FIA_ATD.1/AID
FMT_MTD.1/JCRE	FMT_SMF.1) and (FMT_SMR.1	Yes, FMT_SMF.1, FMT_SMR.1
FMT_MTD.3/JCRE	FMT_MTD.1	Yes, FMT_MTD.1/JCRE
FDP_ACC.1/ LifeCycle	FDP_ACF.1	Yes, FDP_ACF.1/LifeCycle
FDP_ACF.1/LifeCycle	FDP_ACF.1	Yes, FDP_ACC.1/LifeCycle
FMT_MSA.1/LifeCycle	[FDP_ACC.1 l, or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	Yes, FDP_ACC.1/LifeCycle, FMT_SMR.1/CM FMT_SMF.1
FMT_MSA.3/Lifecycle	FMT_MSA.1 FMT_SMR.1	Yes, FMT_MSA.1/Lifecycle FMT_SMR.1/CM
FIA_AFL.1/PIN	FIA_UAU.1	No, rationale in Section 0
FTP_ITC.1/LifeCycle	[FDP_ACC.1, or FDP_IFC.1] FMT_MSA.3	Yes, FDP_ACC.1/LifeCycle FMT_MSA.3/LifeCycle
FAU_SAS.1/SCP	No dependencies	

SFR	Dep.	Met?
FCS_RNG.1/RNG2	No dependencies	
FCS_RNG.1	No dependencies	
FPT_EMSEC.1	No dependencies	
FDP_ACC.2/SecureBox	FDP_ACF.1	Yes, FDP_ACF.1/Secure box
FDP_ACF.1/SecureBox	[FDP_ACC.1 l, or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	Yes, FDP_ACC.1/SecureBox FMT_SMF.1/SecureBox FMT_SMR.1
FMT_MSA.3/SecureBox	FMT_MSA.1 FMT_SMR.1	Yes, FMT_MSA.1/Securebox And FMT_SMR.1
FMT_MSA.1/SecureBox	[FDP_ACC.1 or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	Yes, FDP_ACC.1/SecureBox FMT_SMR.1 FMT_SMF.1/SecureBox
FMT_SMF.1/SecureBox	No dependencies	

For the dependencies of the SFRs taken from [10], see Section 6.3.2 of [10].

6.4.1 Rationale for the Exclusion of Dependencies

The dependency FIA_UID.1 of FMT_SMR.1/Installer is unsupported. This PP 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 PP 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 FIA_UAU.1 of FIA_AFL.1/PIN is unsupported. The TOE implements the firewall access control SFP, based on which access to the object Implementing FIA_AFL.1/PIN is organized.

The dependency FDP_ACF.1/JCRMI of FMT_MSA.3/JCRMI is unsupported. The TOE restricts the access to any subject for access to the RMI functionality, the security attributes that are part of this functionality are not used and therefore no management of security attributes is included

6.5 Security Assurance Requirements Rationale

6.5.1.1 Evaluation Assurance Level Rationale

An assurance requirement of **EAL4** is required for this type of TOE since it is intended to defend against sophisticated attacks. This evaluation assurance level was selected since it is designed to permit a developer to gain maximum assurance from positive security engineering based on good commercial practices. **EAL4** represents a practical level of assurance expected for a commercial grade product.

In order to provide a meaningful level of assurance that the TOE provides an adequate level of defense against such attacks, the evaluators should have access to the low level design and source code. The lowest for which such access is required is **EAL4**.

The assurance level **EAL4** is achievable, since it requires no specialist techniques on the part of the developer.

6.5.1.2 Assurance Augmentations Rationale

Additional assurance requirements are also required due to the definition of the TOE and the intended security level to assure.

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.

This assurance component is a higher hierarchical component to EAL4 (only ALC_DVS.1 is found in EAL4). Due to the nature of the TOE, there is a need to justify the sufficiency of these procedures to protect the confidentiality and the integrity of the TOE.

ALC_DVS.2 has no dependencies.

AVA_VAN.5 Advanced methodical vulnerability analysis

Vulnerability analysis is an assessment to determine whether vulnerabilities identified, during the evaluation of the construction and anticipated operation of the TOE or by other methods (e.g. by flaw hypotheses), could allow users to violate the TSP.

This assurance component is a higher hierarchical component to EAL4 (only AVA_VAN.3 is found in EAL4).

AVA_VAN.5 has dependencies with ADV_ARC.1 “Security architecture description”, ADV_FSP.4 “Complete functional specification”, ADV_TDS.3 “Basic modular design”, ADV_IMP.1 “Implementation representation of the TSF”, AGD_OPE.1 “Operational user guidance”, and AGD_PRE.1 “Preparative procedures”. These components are included in EAL4, and so these dependencies are satisfied.

ASE_TSS.2 TOE summary specification with architectural design summary

The TOE summary specification shall describe how the TOE protects itself against interference and logical tampering, and the TOE summary specification shall describe how the TOE protects itself against bypass.

This assurance component is a higher hierarchical component to EAL4 (only ASE_TSS.1 is found in EAL4). Due to the nature of the TOE, there is a need to explain the architecture in more detail.

6.6 Rationale for Security Requirements from [10]

The rationale for the security requirements for the MIFARE DESFire Emulation can be found in the according Security Target [10]. In Table 33 only the assignment between the objectives and the requirements is done. The justification why the single requirements are fulfilling the objectives is given in [10]. There also the justification for the fulfillment of the dependencies and additional SFR’s supporting the security are given.

Table 33. Assignment: Security Objectives for the TOE – Security Requirements

	FMT_SMR.1[DESFire]	FDP_ACC.1[DESFire]	FDP_ACF.1[DESFire]	FMT_MSA.3[DESFire]	FMT_MSA.1[DESFire]	FMT_SMF.1[DESFire]	FDP_ITC.2[DESFire]	FCS_CKM.4[DESFire]	FMT_MTD.1[DESFire]	FCS_COP.1[DESFire_HW_DES	FCS_COP.1[DESFire_HW_AES	FIA_UID.2[DESFire]	FIA_UAU.2[DESFire]	FIA_UAU.5[DESFire]	FTP_TRP.1[DESFire]	FPT_RPL.1[DESFire]	FPT_TDC.1[DESFire]	FDP_ROL.1[DESFire]
OT.DF_DATA-ACCESS	x	x	x	x	x	x	x	x	x									
OT.DF_AUTHENTICATION										x	x	x	x	x	x	x		
OT.DF_CONFIDENTIALITY											x				x	x		
OT.DF_TYPE-CONSISTENCY																	x	
OT.DF_TRANSACTION																		x

7. TOE summary specification (ASE_TSS)

This section provides a description of the security functions and assurance measures of the TOE that meet the TOE security requirements.

7.1 Security Functionality

The following table provides a list of all security functions.

Table 34. List of all security functions

TOE Security Function	Short Description
SF.AccessControl	enforces the access control
SF.Audit	Audit functionality
SF.CryptoKey	Cryptographic key management
SF.CryptoOperation	Cryptographic operation
SF.I&A	Identification and authentication

TOE Security Function	Short Description
SF.SecureManagement	Secure management of TOE resources
SF.PIN	PIN management
SF.LoadIntegrity	Package integrity check
SF.Transaction	Transaction management
SF.Hardware	TSF of the underlying IC
SF.CryptoLib	TSF of the certified crypto library
SF.DFEmulation	TSF of the MIFARE DESFire Emulation in the underlying IC

7.1.1 SF.AccessControl

This security function ensures the access and information flow control policies of the TOE:

SF.ACC_LCM LIFE CYCLE MANAGEMENT access control SFP (see sections 6.1.13.1 FDP_ACC.1/LifeCycle and 6.1.13.2 FDP_ACF.1/LifeCycle, setting the card life cycle state via a trusted channel (see section 6.1.14.2 FTP_ITC.1/LifeCycle).

SF.ACC_FW FIREWALL access control SFP (see sections 6.1.1.1 FDP_ACC.2/FIREWALL and 6.1.1.2 FDP_ACF.1/FIREWALL)[11]

SF.ACC_IFC JCVM information flow control SFP (see section 6.1.1.3 FDP_IFC.1/JCVM and 6.1.1.4 FDP_IFF.1/JCVM).

SF.ACC_SBX Secure Box access control SFP (see sections 6.1.15.1 FDP_ACC.2/SecureBox and 6.1.15.2 FDP_ACF.1/SecureBox)

SF.ACC_PLI PACKAGE LOADING information flow control SFP (see sections 6.1.9.2 FDP_IFC.2/CM, 6.1.9.3 FDP_IFF.1/CM) for the import of user data (see section 6.1.5.1 FDP_ITC.2/INSTALLER)post issuance loading of applets is done via a trusted channel (see 6.1.9.10 FTP_ITC.1/CM)

SF.ACC_ADE ADEL access control SFP for deleting applets (see sections 6.1.6.1 FDP_ACC.2/ADEL, 6.1.6.2 FDP_ACF.1/ADEL

SF.ACC_RMI JCRMI (Java Card Remote Method Invocation) access control SFP (see sections 6.1.7.1 FDP_ACC.2/JCRMI, 6.1.7.2 FDP_ACF.1/JCRMI)

SF.ACC_EME EXTERNAL MEMORY access control SFP (see sections 6.1.10.1 FDP_ACC.1/EXT_MEM and 6.1.10.2 FDP_ACF.1/EXT_MEM)

It further ensures the management of the necessary security attributes:

SF.ACC_MCL MANAGEMENT CARD LIFE CYCLE: Only S.PACKAGE(CM) is allowed to modify the card life cycle state (see sections 6.1.13.3

[11] Note that the TOE does not support multiple logical channels.

	FMT_MSA.1/LifeCycle, 6.1.9.8 FMT_SMF.1/CM, and 6.1.9.9 FMT_SMR.1/CM).
SF.ACC_MCA	MANAGEMENT CONTEXT and ATTRIBUTES: Only the JCRE (S.JCRE) can modify the the SELECTed applet Context security attribute and can change the list of registered applets' AID (see 6.1.1.6 FMT_MSA.1/JCRE, 6.1.4.4 FMT_MTD.1/JCRE, 6.1.1.11 FMT_SMF.1, 6.1.1.12 FMT_SMR.1). Only the JCVM (S.JCVM) can modify the active context and the active applet security attribute. (see 6.1.1.7 FMT_MSA.1/JCVM, 6.1.1.11 FMT_SMF.1, 6.1.1.12 FMT_SMR.1). Furthermore, only the JCRE can set up the security attribute address space (see 6.1.10.3 FMT_MSA.1/EXT_MEM and 6.1.10.5 FMT_SMF.1/EXT_MEM)
SF.ACC_MRF	Management of roles and functions: Only specified roles are allowed to use specified management functions and security attributes (see 6.1.1.12 FMT_SMR.1, 6.1.9.6 FMT_MSA.1/CM, 6.1.9.8 FMT_SMF.1/CM, 6.1.9.9 FMT_SMR.1/CM, 6.1.15.5 FMT_SMF.1/SecureBox, 6.1.15.4 FMT_MSA.1/SecureBox, 6.1.6.4 FMT_MSA.1/ADEL, 6.1.6.6 FMT_SMF.1/ADEL, 6.1.6.7 FMT_SMR.1/ADEL)
SF.ACC_SVA	SECURE VALUES and ATTRIBUTES: Only secure values are accepted for TSF data and security attributes (see 6.1.1.8 FMT_MSA.2/FIREWALL_JCVM, 6.1.4.5 FMT_MTD.3/JCRE, 6.1.1.11 FMT_SMF.1, 6.1.1.12 FMT_SMR.1, 6.1.9.9 FMT_SMR.1/CM). i. e.: - The Context attribute of a *.JAVAOBJECT must correspond to that of an installed applet or be "JCRE". - An OB.JAVAOBJECT whose Sharing attribute is a JCRE entry point or a global array necessarily has "JCRE" as the value for its Context security attribute. - An OB.JAVAOBJECT whose Sharing attribute value is a global array necessarily has "array of primitive Java Card System type" as a JavaCardClass security attribute's value. - Any OB.JAVAOBJECT whose Sharing attribute value is not "Standard" has a PERSISTENT-LifeTime attribute's value. - Any OB.JAVAOBJECT whose LifeTime attribute value is not PERSISTENT has an array type as JavaCardClass attribute's value.
SF.ACC_RDNOV	Restrictive default non overwriteable values are used for the security attributes (see 6.1.13.4 FMT_MSA.3/LifeCycle, 6.1.1.9 FMT_MSA.3/FIREWALL, 6.1.1.10 FMT_MSA.3/JCVM, 6.1.9.7 FMT_MSA.3/CM, 6.1.6.5 FMT_MSA.3/ADEL)
SF.ACC_RDV	Restrictive default values are used for the security attributes, which can be overwritten (see 6.1.15.3 FMT_MSA.3/SecureBox).
SF.ACC_SDV	The JCRE sets default values when an object or information is created (see 6.1.10.4 FMT_MSA.3/EXT_MEM).

7.1.2 SF.Audit

SF.Audit shall be able to accumulate or combine in monitoring the following auditable events and indicate a potential violation of the TSP:

SF.AUD_AEC	Abnormal environmental conditions (frequency, voltage, temperature), in fulfillment of FAU_ARP.1, and FPT_FLS.1.
SF.AUD_PHT	Physical tampering, in fulfillment of FAU_ARP.1, FPT_FLS.1.
SF.AUD_EFA	EEPROM failure audited by detection of broken EEPROM cells during write operations, in fulfillment of FAU_ARP.1,and FPT_FLS.1.
SF.AUD_CLI	Card life cycle state inconsistency audited through the life cycle checks in all administrative operations and the self test mechanism on start-up, in fulfillment of FAU_ARP.1, and FPT_FLS.1.
SF.AUD_OLI	OS internal life cycle state inconsistency audited through the life cycle checks in all administrative operations (root applet) in fulfillment of FAU_ARP.1, and FPT_FLS.1.
SF.AUD_ALI	Applet life cycle inconsistency, in fulfillment of FAU_ARP.1, and FPT_FLS.1.
SF.AUD_CCS	Corruption of check-summed objects, in fulfillment of FAU_ARP.1, and FPT_FLS.1.
SF.AUD_UOR	Unavailability of resources audited through the object allocation mechanism, in fulfillment of FAU_ARP.1, and FPT_FLS.1.
SF.AUD_AOT	Abortion of a transaction in an unexpected context (see [19] and [20], §7.6.2), in fulfillment of FAU_ARP.1, and FPT_FLS.1.

Based on the events listed above and the following events (also see 6.1.3.1):

SF.AUD_VFJ	Violation of the Firewall or JCVM SFPs, in fulfillment of FAU_ARP.1 and FPT_FLS.1.
SF.AUD_AOF	Array overflow, in fulfillment of FAU_ARP.1, and FPT_FLS.1.
SF.AUD_ORE	Other runtime errors (like uncaught exceptions , CAP file inconsistency, errors in operands of a bytecode, access violations), in fulfillment of FAU_ARP.1, and FPT_FLS.1.
SF.AUD_CDT	Card tearing (unexpected removal of the Card out of the CAD) and power failure, in fulfillment of FAU_ARP.1, and FPT_FLS.1.

SF.Audit shall throw an exception, lock the card session or reinitialize the Java Card System and its data upon detection of one or more of these potential security violations or respond automatically in the specified way (see 6.1.12.3) according to the ST lite [10].

Note: The following reactions by the TOE based on indication of a potential violation of the TSP are possible:

a) Throw an exception

b) Terminate the card (Life cycle state: TERMINATED)

c) Reinitialize the Java Card System (warm reset)

d) responding automatically according to FPT_PHP.3 [10] integrity of the EEPROM and the ROM: The EEPROM is able to correct a 1-bit error within each byte. The

ROM provides a parity check. The EEPROM corrects errors automatically without user interaction, a ROM parity error forces a reset.)

e) Lock the card session (simply stops processing; escape with reset the session/Card tearing)

Based on these types of response/reaction the events listed above will have the following mapping:

Table 35. Response/Reaction on SF.Audit events

Event #	Exception	Terminate card	HW Reset IC or other HW action	Lock card session
Abnormal environmental conditions			X	
Physical tampering	X	X	X	X
EEPROM failure audited		X		
Card Manager life cycle state inconsistency audited through the life cycle checks in all administrative operations		X		
OS internal life cycle				X
Applet life cycle inconsistency		X		
Corruption of check-summed objects	X			X
Unavailability of resources audited through the object allocation mechanism.	X			
Abortion of a transaction in an unexpected context	X			
Violation of the Firewall or JCVM SFPs	X			
Array overflow	X			
Other runtime errors	X	X		X
Card tearing (unexpected removal of the Card out of the CAD) and power failure			X	

7.1.3 SF.CryptoKey

This TSF is responsible for secure cryptographic key management. Cryptographic operation is provided by the following TSF. This TSF provides the following functionality:

SF.CRK_GDE Generation of DES keys with length of 112 and 168 Bit based on random numbers according to AIS 20 [8] class DRG.3 or DRG.3 (see 6.1.2.1 FCS_CKM.1 and 6.1.14.4 FCS_RNG.1 and 6.1.14.5 *FCS_RNG.1/RNG2*).

SF.CRK_GRS Generation of RSA keys with length from 1976 to 2048 Bit based on random numbers according to AIS 20 [8] class DRG.3 (see 6.1.2.1 FCS_CKM.1 and 6.1.14.4 FCS_RNG.1 and 6.1.14.5 *FCS_RNG.1/RNG2*).

SF.CRK_GAE	Generation of AES keys with length of 128, 192, and 256 Bit based on random numbers according to AIS 20 [8] class DRG.3 (6.1.2.1 FCS_CKM.1 and 6.1.14.4 FCS_RNG.1 and 6.1.14.5 *FCS_RNG.1/RNG2*).
SF.CRK_DDE	Distribution of DES keys according to Java Card API [19] or proprietary API [34] (see 6.1.2.2 FCS_CKM.2).
SF.CRK_DRS	Distribution of RSA keys according to Java Card API [19] or proprietary API [34] (see 6.1.2.2 FCS_CKM.2).
SF.CRK_DAE	Distribution of AES keys according to Java Card API [19] or proprietary API [34] (see 6.1.2.2 FCS_CKM.2).
SF.CRK_MOK	Management of DES, AES, RSA, RSA CRT, and EC keys with methods/commands defined in packages javacard.security of Java Card API [19] and proprietary methods defined in [34] (see 6.1.2.3 FCS_CKM.3).
SF.CRK_DOK	Destruction of DES, AES, RSA, RSA CRT, and EC keys by physically overwriting the keys by method clearKey of Java Card API [19] (see 6.1.2.4 FCS_CKM.4).
SF.CRK_GEC	Generation of ECC over GF(p) keys with length from 128 to 320 Bit based on random numbers according to AIS 20 [8] class DRG.2 or DRG.3 (see 6.1.2.1 FCS_CKM.1).
SF.CRK_DEC	Distribution of ECC over GF(p) keys according to Java Card API [19] (see 6.1.2.2 FCS_CKM.2).
SF.CRK_DST	Destruction of session keys by physically overwriting the keys by overwriting them with zeros when explicitly deleted or when the applet is deselected (see 6.1.2.4 FCS_CKM.4)

7.1.4 SF.CryptoOperation

This TSF is responsible for secure cryptographic operation. Cryptographic key management is provided by the previous TSF. This TSF provides the following functionality:

SF.COP_DES	Data encryption and decryption with Triple-DES in ECB/CBC Mode and cryptographic key sizes of 112 and 168 Bit that meets ANSI X9.52-1998 [45] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/TripleDES)
SF.COP_RSA	Data encryption and decryption with RSA and PKCS#1 padding [24]. Key sizes range from 1976 to 2048 Bit (for details, especially on the supported schemes and padding algorithms 6.1.2.5 FCS_COP.1/ RSACipher).
SF.COP_MAC	8 byte MAC generation and verification with Triple-DES in outer CBC Mode and cryptographic key size of 112 and 168 Bit according to ISO 9797-1 [27] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/ DESMAC).
SF.COP_AMC	16 byte MAC generation and verification with AES in CBC Mode and cryptographic key size of 128 Bit according to ISO 9797-1 [27] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/ AESMAC).

SF.COP_AES	Data encryption and decryption with AES in ECB/CBC Mode and cryptographic key sizes of 128, 192, and 256 Bit that meets FIPS 197 [23] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/ AES).
SF.COP_RSI	RSA digital signature generation and verification with SHA-1 and SHA-256 as hash function and cryptographic key sizes from 1976 to 2048 Bit according to ISO 9796-2 [26] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/RSASignatureISO9796).
SF.COP_RSP	RSA digital signature generation and verification with SHA-1 and SHA-256 as hash function and cryptographic key sizes from 1976 to 2048 Bit according to PKCS#1 [24] (for details, especially on the supported schemes and padding algorithms 6.1.2.5 FCS_COP.1/ RSASignaturePKCS#1).
SF.COP_RSS	RSA digital signature generation and verification with SHA-1, SHA-224 and SHA-256 as hash function and cryptographic key sizes from 1976 to 2048 Bit according to PKCS#1_PSS [24] (for details, especially on the supported schemes and padding algorithms 6.1.2.5 FCS_COP.1/ RSASignaturePKCS#1_PSS)).
SF.COP_HS1	Secure hash computation with SHA-1 according to FIPS 180-3 [29] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/SHA-1).
SF.COP_RNG	Random number generation according to AIS 20 [8] class DRG.3 (see 6.1.14.4 FCS_RNG.1).
SF.COP_RNG2	Random number generation according to AIS 20 [8] class DRG.2 (see 6.1.14.5 FCS_RNG.1/RNG2).
SF.COP_ESI	EC Digital signature generation and verification with SHA-1, SHA-224, and SHA-256 as hash functions and cryptographic key sizes from 128 to 320 Bit according to ISO14888-3 [28] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/ECSignature).
SF.COP_HS2	Secure hash computation with SHA-224 according to FIPS 180-3 [29] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/ SHA-224).
SF.COP_HS5	Secure hash computation with SHA-256 according to FIPS 180-3 [29] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/ SHA-256).
SF.COP_SMI	Secure Messaging functionality for ICAO – either encryption and decryption with Triple-DES in CBC mode and cryptographic key size of 112 bit FIPS 46-3 [22] , as well as message authentication code with Retail MAC and cryptographic key size of 112 bit according to ISO 9797-1 [27] or encryption and decryption with AES in CBC mode (see FIPS 197 [23]) and message authentication wit AES-CMAC (NIST 800-38B) both with cryptographic key sizes of 128, 192, or 256 (for details, especially on the supported schemes and padding

algorithms see 6.1.2.5 FCS_COP.1/AES and FCS_COP.1/AES_CMAC).[12]

SF.COP_DHK — Diffie-Hellman key agreement with ECC over GF(p) and RSA supporting cryptographic key sizes from 128 to 320 bit (for ECC) and from 1976 to 2048 bit (for RSA) according to ISO 11770-3 [25] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/DHKeyExchange).

SF.COP_SPA — Secure point addition in accordance with the specified cryptographic algorithm ECC over GF(p) and cryptographic key sizes 128 to 320 Bit according to ISO14888-3 [28] (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/ECAdd).

SF.COP_AEC — AES-CMAC computation according to NIST 800-38B [32] with cryptographic key length of 128, 192, and 256 (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/AES_CMAC

SF.COP_TDC — TDES-CMAC computation according to NIST 800-38B [32] with cryptographic key length of 112 bit (for details, especially on the supported schemes and padding algorithms see 6.1.2.5 FCS_COP.1/TDES_CMAC).

7.1.5 SF.I&A

The TSF provides the following functionality with respect to card manager (administrator) authentication:

SF.I&A_CRM — The TSF provides a challenge-response mechanism for card manager authentication and ensures that the session authentication data cannot be reused. After successful authentication, a trusted channel that is protected in integrity and confidentiality is established (6.1.14.2 FTP_ITC.1/LifeCycle).

SF.I&A_UCA — The TSF blocks the card when 66 consecutive unsuccessful card manager authentication attempts via secure messaging using D.APP_KEY occur (see 6.1.9.3 FDP_IFF.1/CM).

SF.I&A_EBA — Package execution is possible before authentication (*6.1.9.5* FIA_UID.1/CM).

7.1.6 SF.SecureManagment

The TSF provide a secure management of TOE resources:

SF.SMG_AID — The TSF maintain a unique AID and version number for each package, the AID of each registered applet, and whether a registered applet is currently selected for execution ([21], §6.5) (see 6.1.4.1 FIA_ATD.1/AID, 6.1.4.2 FIA_UID.2/AID and 6.1.4.3 FIA_USB.1/AID).

SF.SMG_UOO — The TSF ensures that packages are unable to observe operations on secret keys and PIN codes by other subjects (see 6.1.3.3 FPR_UNO.1).

SF.SMG_MIE — The TSF monitors user data D.APP_CODE, D.APP_I_DATA, D.PIN, D.APP_KEYs for integrity errors. If an error occurs for D.APP_KEYs

12. Other secure messaging functionality is part of the SF.COP_DES and SF.COP_MAC. Key destruction for ICAO functionality is part of SF.CRK_DST.

or D.PIN, the TSF maintain a secure state (lock card session). If an error occurs for D.APP_CODE or D.APP_I_DATA, a SecurityException is thrown (see 6.1.3.2 FDP_SDI.2).

SF.SMG_PIU — The TSF makes any previous information content of a resource unavailable upon (see 6.1.1.5 FDP_RIP.1/OBJECTS, 6.1.2.7 FDP_RIP.1/APDU, 6.1.2.8 FDP_RIP.1/bArray, 6.1.2.10 FDP_RIP.1/TRANSIENT, 6.1.2.6 FDP_RIP.1/ABORT, 6.1.2.9 FDP_RIP.1/KEYS, 6.1.6.3 FDP_RIP.1/ADEL, 6.1.8.1 FDP_RIP.1/ODEL):

- allocation of class instances, arrays, and the APDU buffer,
- de-allocation of bArray object, any transient object, any reference to an object instance created during an aborted transaction, and cryptographic buffer (D.CRYPTO).
- de-allocation of applets and objects

SF.SMG_NSC — NO SIDE-CHANNEL: The TSF ensures that during command execution there are no usable variations in power consumption (measurable at e. g. electrical contacts) or timing (measurable at e. g. electrical contacts) that might disclose cryptographic keys or PINs.[13] All functions of SF.CryptoOperation except with SHA are resistant to side-channel attacks (e.g. timing attack, SPA, DPA, DFA, EMA, DEMA) (see 6.1.14.6 FPT_EMSEC.1).

SF.SMG_CAP — CAP files, the bytecode and its data arguments are consistently interpreted using the following rules (see 6.1.3.5 FPT_TDC.1):

a. The virtual machine specification [21];
b. Reference export files;
c. The ISO 7816-6 rules;
d. The EMV specification.

SF.SMG_SSI — The TSF ensures a secure state when the installer fails to install or load a package or applet (see 6.1.5.3 FPT_FLS.1/Installer, 6.1.5.4 FPT_RCV.3/Installer)

SF.SMG_AOD — The TSF ensures a secure state when the applet or object deletion fails (see 6.1.6.8 FPT_FLS.1/ADEL, 6.1.8.2 FPT_FLS.1/ODEL)

7.1.7 SF.PIN

The TSF provides the following functionality with respect to user authentication with the global PIN (D.PIN):

SF.PIN_NUP — The maximum possible number of consecutive unsuccessful PIN-authentication attempts is user configurable number from 1 to 127. (see 6.1.14.1 FIA_AFL.1/PIN)

SF.PIN_PAB — When this number has been met or surpassed, the PIN-authentication is blocked (FIA_AFL.1/PIN).

SF.PIN_CBI — Only the following commands are allowed, before successful identification (see 6.1.9.5 FIA_UID.1/CM):

[13] Note: All measures described in guidance of the underlying hardware platform concerning power consumption and timing will be taken into account for the TOE development.

- *Get Data* with objects:	ISD DATA [ISSUER IDENTIFICATION NUMBER], ISD DATA [CARD IMAGE NUMBER], PLATFORM DATA [CARD RECOGNITION DATA], ISD DATA [KEY INFORMATION TEMPLATE], ISD DATA [SCP INFORMATION], PLATFORM DATA [MANUFACTURING]
- *Select Applet*	
- *Initialize Update* with object:	APDU BUFFER
- *External Authenticate* with object:	APDU BUFFER

7.1.8 SF.LoadIntegrity

SF.LIT_OIP The TSF ensures the origin and the integrity of a received package (see secions 6.1.9.1 FCO_NRO.2/CM and 6.1.9.4 FDP_UIT.1/CM). The algorithm used for providing integrity protection is RSASSA-PKCS1-v1_5 [24] with a key length of 2048 bit according to [15].

7.1.9 SF.Transaction

SF.TRA_PRO The TSF permits the rollback of operations OP.JAVA, OP.CREATE on objects OB.JAVAOBJECTs. These operations can be rolled back within the calls: select(), deselect(), process() or install(), notwithstanding the restrictions given in Java Card Runtime Environment [20], §7.7, within the bounds of the Commit Capacity ([20], §7.8), and those described in Java Card API [19]. (see 6.1.2.11 FDP_ROL.1/FIREWALL).

7.1.10 SF.Hardware

The certified hardware (part of the TOE) features the following TSF. The exact formulation can be found in the hardware security target [10]:

SF.HW_RNG Random Number Generator (F.RNG) used for SF.COP_RNG (see 6.1.14.4 *FCS_RNG.1*)) and . SF.COP_RNG2 (see 6.1.14.5 *FCS_RNG.1/RNG2)*.

SF.HW_TDC Triple-DES Co-processor (F.HW_DES) used for SF.CYL_SDE and SF.COP_RNG and SF.COP_RNG2 (see 6.1.2.5 FCS_COP.1/TripleDES, FCS_COP.1/DESMAC, FCS_COP.1/TDES_CMAC and 6.1.14.4 FCS_RNG.1 and 6.1.14.5 *FCS_RNG.1/RNG2*).

SF.HW_AEC AES Co-processor (F.HW_AES) used for SF.COP_AES (see 6.1.2.5 FCS_COP.1/AES, FCS_COP.1/AESMAC, and FCS_COP.1/AES_CMAC).

SF.HW_COC Control of Operating Conditions (F.OPC) (see 6.1.12.1 FPT_FLS.1/SCP, 6.1.12.2 FRU_FLT.2/SCP).

SF.HW_PPM Protection against Physical Manipulation (F.PHY) (see 6.2.2.5 FCS_COP.1/TripleDES and FCS_COP.1/AES, 6.1.14.4 FCS_RNG.1, 6.1.14.5 *FCS_RNG.1/RNG2,* 6.1.2.5 FPT_FLS.1/SCP, 6.1.12.3

FPT_PHP.3/SCP, 6.1.12.2 FRU_FLT.2/SCP, 6.1.14.3 FAU_SAS.1/SCP, , 6.1.12.4 FDP_ACC.1/SCP, 6.1.12.5 FDP_ACF.1/SCP, and 6.1.12.6 FMT_MSA.3/SCP).

SF.HW_LOG — Logical Protection (F.LOG) (see 6.1.14.6 FPT_EMSEC.1).

SF.HW_PMC — Protection of Mode Control (F.COMP) (see 6.1.14.3 FAU_SAS.1/SCP).

SF.HW_MACC — Memory Access Control (F.MEM_ACC). The functionality of the hardware is used for the MIFARE firewall (see 6.1.12.4 FDP_ACC.1/SCP, 6.1.12.5 FDP_ACF.1/SCP, and 6.1.12.6 FMT_MSA.3/SCP), and to implement the Secure Box (see 6.1.15.1 FDP_ACC.2/SecureBox, 6.1.15.2 FDP_ACF.1/SecureBox)

SF.HW_RAC — Special Function Register Access Control (F.SFR_ACC). The functionality of the hardware is used by the TOE to implement the Secure Box (see 6.1.15.1 FDP_ACC.2/SecureBox, 6.1.15.2 FDP_ACF.1/SecureBox).

7.1.11 SF.CryptoLib

The certified cryptographic library (part of the TOE) features the following TSF. The exact formulation can be found in the crypto library security target **[9]**:

SF.CYL_SAE — Software AES (F.AES) based on F.HW_AES. The functionality of the cryptographic library is not used by the TOE and not exposed at external interfaces of the composite TOE.

SF.CYL_SDE — Software DES (F.DES) based on SF.HW_DES used for SF.COP_DES, SF.COP_MAC, SF.COP_SMI, and SF.COP_TDC (see 6.1.2.5 FCS_COP.1/TripleDES, FCS_COP.1/DESMAC, , FCS_COP.1/TDES_CMAC).

SF.CYL_RSA — RSA encryption (F.RSA_encrypt). The functionality of the cryptographic library is not used by the TOE and not exposed at external interfaces of the composite TOE.

SF.CYL_RSS — RSA signing (F.RSA_sign). The functionality of the cryptographic library is not used by the TOE and not exposed at external interfaces of the composite TOE.

SF.CYL_RKC — RSA public key computation (F.RSA_public). The functionality of the cryptographic library is not used by the TOE and not exposed at external interfaces of the composite TOE.

SF.CYL_ECS — ECC Signature Generation and Signature Verification (F.ECC_GF_p_ECDSA) used for SF.COP_ESI (see 6.1.2.5 FCS_COP.1/ECSignature).

SF.CYL_DHK — Diffie-Hellman Key Exchange (F.ECC_GF_p_DH_KeyExch) used for SF.COP_DHK (see 6.1.2.5 FCS_COP.1/DHKeyExchange).

SF.CYL_RKG — RSA Key Pair Generation (F.RSA_KeyGen). The functionality of the cryptographic library is not used by the TOE and not exposed at external interfaces of the composite TOE.

SF.CYL_EKG EC Key Generation (F.ECC_GF_p_KeyGen) used for SF.CRK_GEC (see 6.1.2.1 FCS_CKM.1). according to ISO/IEC 15946-1 [18] and [19].

[SF.CYL_CSH	Compute the Secure Hash Algorithms (F.SHA) used for SF.COP_HS1, SF.COP_HS2, and SF.COP_HS5 (see 6.1.2.5 FCS_COP.1/SHA-1, 6.1.2.5 FCS_COP.1/SHA-224, 6.1.2.5 FCS_COP.1/SHA-256)
SF.CYL_SPR	Software pseudo random number generator (F.RNG_Access). The functionality of the cryptographic library is not used by the TOE and not exposed at external interfaces of the composite TOE.
SF.CYL_CMA	Clear memory areas used by the Crypto Library after usage (F.Object_Reuse) is used for SF.CYL_SDE, SF.CYL_ECS, SF.CYL_DHK and SF.CYL_EKG (see 6.1.2.9. FDP_RIP.1/Keys)
SF.CYL_LOG	Logical Protection (F.LOG) extends F.LOG of the Hardware and is used for SF.CYL_SDE, SF.CYL_ECS, SF.CYL_DHK, SF.CYL_EKG, [18] and [19]. SF.CYL_CSH and SF.CYL_MCP (see 6.1.14.6 FPT_EMSEC.1, and 6.1.12.1 FPT_FLS.1/SCP).
SF.CYL_CKD	Cryptographic Key Destruction. The functionality of the cryptographic library is not used by the TOE and not exposed at external interfaces of the composite TOE.
SF.CYL_EPA	ECC Point addition (FCS_COP.1[ECC_ADD] in F.ECC_GF_p_ECDSA) used for SF.COP_SPA (see 6.1.2.5 FCS_COP.1/ECAdd).
SF.CYL_MCP	Memory copy in a manner protected against side channel attacks (F.COPY)

7.1.12 SF.DFEmulation

SF.DFE_AUT	The MIFARE DESFire Emulation authentication mechanism provides an access control mechanism to the objects and security attributes that are part of the Access Control Policy for the MIFARE DESFire Emulation. See FMT_SMR.1[DESFire], FDP_ACC.1[DESFire], FDP_ACF.1[DESFire], FMT_MSA.3[DESFire], FMT_MSA.1[DESFire], FMT_SMF.1[DESFire], FDP_ITC.2[DESFire], FCS_CKM.4[DESFire], and FMT_MTD.1[DESFire] in Table 27.
SF.DFE_ACC	The Access control service identifies the user (Administrator, Application Manager, Application User, Originality Key User or Everybody) by the currently selected context (DESFire card level or specific application) and the key number selected for the authentication. See FCS_COP.1[DESFire_HW_DES], FCS_COP.1[DESFire_HW_AES], FIA_UID.2[DESFire], FIA_UAU.2[DESFire], FIA_UAU.5[DESFire], FTP_TRP.1[DESFire], and FPT_RPL.1[DESFire] in Table 27.
SF.DFE_CFI	The confidentiality service provides a mechanism to protect the communication against eavesdropping. In order to do this the communication can be encrypted. See FCS_COP.1[DESFire_HW_DES], FTP_TRP.1[DESFire], and FPT_RPL.1[DESFire] in Table 27.
SF.DFE_TYP	The type consistency check ensures the type consistency of the file types stored by the DESFire EV1 Software. It ensures that values cannot over- or underflow. See FPT_TDC.1[DESFire] in Table 27.

SF.DFE_TRA — The transaction service is always active for the respective file types. This means that for every modifying operation with a backup file an explicit commit request must be issued in order to let the modifications take effect. See FDP_ROL.1[DESFire] in Table 27.

7.2 Logical Protection

The following chapter gives a short overview of the logical protection mechanisms implemented in the OS.

Applet firewall — The applet firewall is used to separate the different applications and their data from each other and from the Java Card OS.

MMU — The hardware based Memory Management Unit is used to separate native code which is executed as a library inside the Secure Box feature from the OS. It limits and controls the access of this native code to all recourses (ROM, RAM, non volatile memory, and SFRs) of the hardware.

Transaction Mechanism — This mechanism ensures that in case of a tearing event (sudden loss of power) the operating system as well as the executing applet is kept in a consistent state. This means that all operations are performed entirely or get rolled back at next power up cycle.

Secure Channel — The OS provides secure channels for communication with off card systems to ensure the confidentiality, integrity, and authenticity of the transferred data.

Authentication Retry Counter — The OS limits the number of unsuccessful authentications to a predefined number.

7.3 Physical Protection

In the course of this chapter an overview of mechanisms to protect against physical manipulation is given.

Protected Values — For security relevant values the OS uses values coded in a redundant manner to allow the detection of manipulations.

Secure Copy — It is a mechanism to securely move data from one location to another. In particular, this mechanism protects against leakage of data through side-channels.

Clear Memory — Memory areas containing sensitive data are cleared after usage. This is also supported by the used crypto library which also clears all used memory areas after usage.

Secure Compare — It is a mechanism to securely compare data. In particular, this mechanism protects against leakage of data through side-channels and hardens fault attacks.

Secure Boolean Conversion — It is a mechanism to securely cast Boolean variables into a Secure Value.

Self Test — The OS runs a suite of self tests including tests of RNG and consistency checks on configuration data

Attack Counter	The system maintains a attack counter which counts the number of detected attacks and ensures the termination of the card when the threshold value is reached.
Secure AES	The software part of the AES implementation is done in a way to support the protection against DPA, DFA and timing attacks
Secure RSA	The implementation of the RSA algorithm is done in a way which offers protection against DPA, DFA, and timing attacks.
Secure DES	The software part of the DES implementation is done in a way to support the protection against DPA, DFA and timing attacks (the OS ads here additional features to protect from DFA, DPA measures are part of the certified platform)
Secure ECC	The implementation of the ECC algorithm is done in a way which offers protection against DPA, DFA, and timing attacks (the implementation is fully done in the certified platform).

7.4 Security Features of Hardware

This section gives a short overview of the security features of the underlying CC certified hardware which support the overall security architecture of the TOE.

Coprocessor	The hardware features cryptographic coprocessors for AES, DES and a coprocessor for PKI with protection mechanisms against DPA, DFA and timing attacks
Security Sensors	Enhanced security sensors for clock frequency range, low and high temperature sensor, supply voltage sensors Single Fault Injection (SFI) attack detection, Light sensors (included integrated memory light sensor functionality)
Secure Fetch	Implementation of protection of the code fetch from ROM, RAM and EEPROM
Memory security	Security of memory is based on encryption and physical measures for RAM, EEPROM and ROM
Memory Management Unit (MMU)	The in hardware implemented MMU is able to perform access control to all types of memory and the special functions registers depending on the current CPU mode.
Secure Lock of Testmode	The testmode of the hardware is disabled after the production test. The hardware prevents that this mode can be enabled or reached afterwards to disclose or anipulate TSF data.

8. Bibliography

[1] Common Criteria for Information Technology Security Evaluation, Part 1, Version 3.1, Revision 4, September 2012

[2] Common Criteria for Information Technology Security Evaluation, Part 2, Version 3.1, Revision 4, September 2012

[3] Common Criteria for Information Technology Security Evaluation, Part 3, Version 3.1, Revision 4, September 2012

[4] Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 4, September 2012, CCMB-2012-09-004

[5] Java Card System - Open Configuration Protection Profile, Version 2.6, Certified by ANSSI, the French Certification Body April, 19th 2010

[6] Smartcard IC Platform Protection Profile (SSVG-PP), Version 1.0, June 2007; registered and certified by (BSI) under the reference BSI-PP-0035-2007

[7] Embedded Software for Smart Secure Devices Protection Profile, v1.0, November 27th 2009, ANSSI.

[8] Anwendungshinweise und Interpretationen zum Schema, AIS 20: Funktionalitaetsklassen und Evaluationsmethodologie fuer deterministische Zufallszahlengeneratoren, Version 2.1, 02.12.2011, Bundesamt fuer Sicherheit in der Informationstechnik

[9] Crypto Library V2.7/V2.9 on SmartMX P5Cx081/ CD041/ CD021/ CD016 V1D, Security Target, Rev. 1.3, 08 May 2013, BSI-DSZ-CC-0864

[10] NXP Secure Smart Card Controllers P5CD016V1D / P5CD021V1D / P5CD041V1D / P5Cx081V1D, Security Target Lite, NXP Semiconductors, Revision 1.1, BSI-DSZ-CC-0707, 24. October 2011

[11] Guidance, Delivery and Operation Manual, NXP Secure Smart Card Controllers P5CD016V1D/P5CD021V1D/P5CD041V1D/P5Cx081V1D, NXP Semiconductors, BSI-DSZ-CC-0707, Doc. ID 208730

[12] Virtual Machine Specification, Java Card(tm) Platform, Version 2.2.2, March 2006, Sun Microsystems

[13] The Java Virtual Machine Specification, Lindholm, Yellin. ISBN 0-201-43294-3

[14] The Java Language Specification, Gosling, Joy and Steele, ISBN 0-201-63451-1

[15] GlobalPlatform Card Specification, Version 2.2.1, January 2011

[16] Product Data Sheet addendum P5CD016/021/041 V1D and P5Cx081 V1D with MIFARE DESFire EV1 OS, NXP Semiconductors, Doc.No. 192730

[17] Hardware data sheet, JCOP V2.4.2 Revision 3 J3E081 DF secure smart card controller, Doc.No. 258702, 26 March 2013

[18] ISO/IEC 15946-1: Information technology - Security techniques - Cryptographic techniques based on elliptic curves - Part 1: General, 2008

[19] Application Programming Interface Java Card(tm) Platform, Version 3.0.1, Classic Edition, May 2009, Sun Microsystems, Inc.

[20] Runtime Environment Specification Java Card(tm) Platform, Version 3.0.1 Classic Edition, May 2009, Sun Microsystems, Inc.

[21] Virtual Machine Specification Java Card(tm) Platform, Version 3.0.1 Classic Edition, May 2009, Sun Microsystems, Inc.

[22] FIPS PUB 46-3: FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, DATA ENCRYPTION STANDARD (DES), Reaffirmed 1999 October 25, U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology

[23] FIPS PUB 197: Federal Information Processing Standards Publication 197, Announcing the ADVANCED ENCRYPTION STANDARD (AES), November 26, 2001

[24] PKCS1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 2002

[25] ISO/IEC 11770 Part 3: Information technology - Security techniques - Key management: Mechanisms using asymmetric techniques

[26] ISO/IEC 9796-2:2002: Information technology -Security techniques -Digital signature schemes giving message recovery -Part 2: Integer factorization based mechanisms

[27] ISO/IEC 9797-1:1999: Information technology - Security techniques - Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher

[28] ISO/IEC 14888-3: Information technology, Security techniques, Digital signatures with appendix, Part 3: Discrete logarithm based mechanisms, 2008

[29] FIPS PUB 180-3, Secure Hash Standard, Federal Information Processing Standards Publication, October 2008, US Department of Commerce/National Institute of Standards and Technology

[30] FIPS PUB 180-1: FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, SECURE HASH STANDARD, 1995 April 17

[31] Common Criteria Protection Profile - Machine Readable Travel Document with "ICAO Application", Basic Access Control, Version 1.0, 18.08.2005 (registered at BSI under Registration number BSI-PP-0017)

[32] NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Computer Security Division Information Technology Laboratory National Institute of Standards and Technology Gaithersburg, MD 20899-8930, May 2005

[33] JCOP V2.4.2 Revision 3 Secure Smart Card Controller User Manual, Rev. 0.6, 16th July 2013, Docstore ID 258406

[34] JCOP V2.4.2 Revision 3 Secure Smart Card Controller Administrator Manual, Rev. 0.5, 16th July 2013, Docstore ID: 258605

[35] Anwendungshinweise und Interpretationen zum Schema, AIS 31: Funktionalitaetsklassen und Evaluationsmethodologie fuer physikalische Zufallszahlengeneratoren, Version 1, 25.09.2001, Bundesamt fuer Sicherheit in der Informationstechnik

[36] UM SecureBox JCOP V2.4.2, 221134 Rev 3.4,

[37] RFC 5639 ECC Brainpool Standard Curves & Curve Generation, March 2010 — available at: http://tools.ietf.org/html/rfc5639

[38] FIPS PUB 186-2, U.S. DEPARTMENT OF COMMERCE — National Institute of Standards and Technology, Issued January 27, 2000

[39] DES MODES OF OPERATION - FIPS PUB 81, Federal Information Processing Standards Publication 81, December 2, 1980

[40] Recommendation for Block Cipher Modes of Operation - Methods and Techniques- NIST Special Publication 800-38A, National Institute of Standards and Technology, 2001

[41] SECURE HASH STANDARD, Federal Information Processing Standards Publication 180-4, October, 2008

[42] Kryptographische Verfahren: Empfehlungen und Schlüssellängen, BSI - Technische Richtlinie BSI TR-02102 v2.0, 09.01.2013

[43] Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, NIST Special Publication 800-67 Revision 1 - National Institute of Standards and Technology - January 2012

[44] FIPS PUB 81, DES modes of operation, Federal Information Processing Standards Publication, December 2nd , 1980, US Department of Commerce/National Institute of Standards and Technology

[45] American National Standard: Triple data encryption algorithm modes of operation, ANSI X9.52, November 9th , 1998

[46] [FIPS PUB 197: Federal Information Processing Standards Publication 197, Announcing the ADVANCED ENCRYPTION STANDARD (AES), November 26, 2001]

[47] [Digital Signature Standard (DSS) - FIPS PUB 186-3, FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, June, 2009

[48] [RFC 5639 ECC Brainpool Standard Curves & Curve Generation, March 2010 — available at: http://tools.ietf.org/html/rfc5639]

[49] [ISO/IEC 14888-3: Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms, 2008

[50] [PKCS #3: Diffie-Hellman Key-Agreement Standard, RSA Laboratories Technical Note Version 1.4, Revised November 1, 1993 1, 1993

[51] Guidance, Delivery and Operation Manual, NXP Secure Smart Card Controllers P5CD016V1D/P5CD021V1D/P5CD041V1D/P5Cx081V1D, NXP Semiconductors, BSI-DSZ-CC-0707, Doc. ID 208730

9. Legal information

9.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

9.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

9.3 Licenses

ICs with DPA Countermeasures functionality

NXP ICs containing functionality implementing countermeasures to Differential Power Analysis and Simple Power Analysis are produced and sold under applicable license from Cryptography Research, Inc.

9.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

9.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP B.V.

10. List of figures

11. List of tables

12. Contents

Date of release: 13 August 2013

Please be aware that important notices concerning this document and the product(s) described herein, have been included in the section 'Legal information'.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an please send an email to: salesaddresses@nxp.com

Date of release: 13 August 2013

Date of release: 13 August 2013

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an please send an email to: salesaddresses@nxp.com

Date of release: 13 August 2013