NXP J3A080 and J2A080 Secure Smart Card Controller Rev. 3

Security Target Lite

Rev. 01.02 — 08 December 2010 **Evaluation documentation**

Approved **PUBLIC INFORMATION**

BSI-DSZ-CC-0674

Document information

Info	Content
Keywords	JCOP, ST, Security Target Lite
Abstract	This is the Security Target Lite for JCOP v2.4.1 Revision 3. It defines the contract for the certification according to Common Criteria.

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* **applet**s *and* *package*s. *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, *package*s' *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 *APDU*s. These are encapsulated in Java Card System by the **`javacard.framework.APDU`** *class* ([7]). *APDU*s 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 file*s 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 *class*es that can be installed on a device and used to execute the *package*'s *class*es 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 ([8] Glossary).
Default applet	The applet that is selected after a card reset ([8], §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 *package*s (*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 ([8], §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.
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
O.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 *class*es and *interface*s. 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 *class*es, for external linking purposes; export files are not used directly in a Java Card virtual machine) and *CAP file*s.
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
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 *package*s. 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

This Security Target Lite has been derived from the full Security Target Rev. 1.02, 8th December 2010.

Table 1. ST reference and TOE reference

Title	NXP J3A080 and J2A080 Secure Smart Card Controller Rev. 3 Security Target
Version	Rev. 01.02
Date	08 December 2010
Author(s)	NXP Semiconductors
Developer	NXP Semiconductors
Product Type	Java Card
TOE name/version	NXP J3A080 and J2A080 Secure Smart Card Controller Rev. 3
Certification ID	BSI-DSZ-CC-0674
TOE hardware	P5CD080V0B, P5CC080V0B
CC used	Common Criteria for Information Technology Security Evaluation Version 3.1, Revision 3, July 2009 (Part 1, Part 2 and Part 3)

1.2 TOE overview

This document details the security target for **NXP J3A080 and J2A080 Secure Smart Card Controller Rev. 3** (also named **JCOP 2.4.1 R3**). It is based on the following protection profile:

- Java Card System - Minimal Configuration Protection Profile, Version 1.1, May 2006 [1]

The ST fulfils all requirements of [1]. This ST even chooses a hierarchically higher EAL and hierarchically higher augmentation than the protection profile. The ST selects EAL5, augmented by ALC_DVS.2 and AVA_VAN.5.

The basis for this composite evaluation is the composite evaluation of the hardware plus the cryptographic library. This has been certified by BSI (BSI-DSZ-CC-0709). The corresponding Security Target Lite is [22]. No Maintence Report is applicable.

The hardware platforms P5CD080V0B and P5CC080V0B alone (i.e. without the cryptographic library) are certified by BSI (BSI-DSZ-CC-0410).The corresponding Security Target Lite is [5] and the applicable maintenance report is [6]. The hardware is compliant to the following protection profile:

- Smartcard IC Platform Protection Profile, Version 1.0, July 2001 [4]

For the P5CD080V0B 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 [5]):

- “MIFARE Emulation = A” in which MIFARE interface is disabled.
- “MIFARE Emulation = B1” in which MIFARE interface is enabled and 1KB MIFARE EEPROM memory is reserved
- “MIFARE Emulation = B4” in which MIFARE interface is enabled and 4KB MIFARE EEPROM memory is reserved

For the P5CC080V0B hardware of this TOE only one configuration exists. This is equivalent to “MIFARE Emulation = A” of P5CD080V0B.

From [4] relevant requirements for the hardware platform were taken. The relevant requirements for the Java Card functionality were taken from [1].

JCOP 2.4.1 R3 is based on Java Card 2.2.2 and Global Platform 2.1.1 industry standards. It implements high security mechanisms and supports:

Different communication protocols:

- T=0
- T=1
- T=CL (contact-less) (available on J3A080, not available on J2A080)

Cryptographic algorithms and functionality:

- 3DES
- AES (Advanced Encryption Standard)
- RSA
- SHA-1, SHA-224, SHA-256
- ECC over GF(p)
- Random number generation

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 is the JCOP 2.4.1 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), and
- Native MIFARE application (physically present but logically disabled in minor configuration “MIFARE Emulation = A” and logically enabled in the minor configurations “MIFARE Emulation = B1” and “MIFARE Emulation = B4” (see section 2.2.5 of the HW Security Target [5]))

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

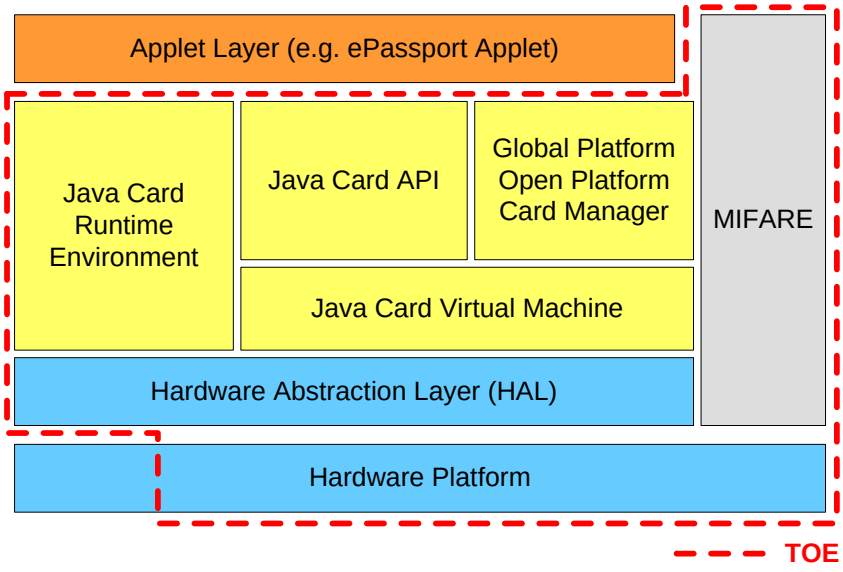


Fig 1. JCOP Architecture

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, but 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. Therefore, some parts of the Hardware Platform are not part of the TOE.

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 [9] 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 2.1.1 [10] and is responsible for the management of applets in the card. No post-issuance loading of applets is allowed for the present TOE. For more details of the Java card functionality see Section 1.3.3.

The native application MIFARE (grey box in Fig 1) is only available in the Minor Configuration options “MIFARE Emulation = B1” and “MIFARE Emulation = B4”. In the Minor Configuration option “MIFARE Emulation = A”, the grey box is not available in the hardware.

The basis for this composite evaluation is the composite evaluation of the hardware plus the cryptographic library. This has been certified by BSI (BSI-DSZ-CC-0709). The corresponding Security Target Lite is [22]. The applicable maintenance report is [28].

The hardware platforms P5CD080V0B and P5CC080V0B alone (i.e. without the cryptographic library) are certified by BSI ((BSI-DSZ-CC-0410).The corresponding Security Target Lite is [5] and is compliant to the following protection profile:

- Smartcard IC Platform Protection Profile, Version 1.0, July 2001 [4].

It is certified to EAL5 augmented by ALC_DVS.2, AVA_MSU.3, and AVA_VLA.4 with all minor configuration options as defined in section 2.2.5 of the HW Security Target [5]. For P5CD080V0B the minor configuration options “MIFARE Emulation = A”, “MIFARE Emulation = B1” and “MIFARE Emulation = B4” can freely be chosen. For P5CC080V0B only “MIFARE Emulation = A” is available in the hardware.

The Java card is based on Java Card 2.2.2 and GlobalPlatform 2.1.1 industry standards. The following features comprise the logical scope of the TOE:

- 5 different communication protocols:
    a. ISO 7816 T=1 direct convention
    b. ISO 7816 T=0 direct convention
    c. ISO 7816 T=1 inverse convention
    d. ISO 7816 T=0 inverse convention
    e. ISO 14443 T=CL (contact-less) (available on J3A080, not available on J2A080)
- Cryptographic algorithms and functionality:
    a. 3DES (112 and 168 bit keys) for en-/decryption (CBC and ECB) and MAC generation and verification (Retail-MAC and CBC-MAC)
    b. AES (Advanced Encryption Standard) with key length of 128, 192, and 256 Bit for en-/decryption (CBC and ECB)
    c. RSA and RSA CRT (1280 up to 2048 bits keys) for en-/decryption and signature generation and verification[1]
    d. RSA CRT key generation (1280 up to 2048 bits keys) in a secured environment
    e. SHA-1, SHA-224, and SHA-256 hash algorithm
    f. ECC over GF(p) for key length between 192 and 320 bits [2]
    g. Random number generation according to class K3 of AIS 20 [21]**.**
    h. ECC over GF(p) point addition with cryptographic key length between 192 and 320 bits.
- Java Card 2.2.2 functionality:
    a. Garbage Collection fully implemented with complete memory reclamation incl. compactification

[1] As of 2011 it is recommended to use a minimum key length for signatures of 1976 bit

[2] As of 2011 it is recommended to use a minimum key length for signatures of 224 bit

b. Support for Extended Length APDUs

- GlobalPlatform 2.1.1 functionality:
  a. CVM Management (Global PIN) fully implemented: all described APDU and API interfaces for this feature are present
  b. Secure Channel Protocol (SCP01, and SCP02) is supported
- Proprietary BAC Accelerator Interface, secure massaging API of JCOP 2.4.1 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.
- Functionality as defined in the JC PP minimal configuration (i.e. no post-issuance installation and deletion of applets, packages and objects, no RMI, no logical channels, no on-card Bytecode verification)
- Card manager functionality for pre-issuance loading and management of packages and applets.
- MIFARE application accesible via contactless interface and via Java Card API (available on J3A080 in Configuartion B1 or B4, not available on J3A080 in Configuartion A, not available on J2A080)

The following functionality of the smart card platform is not used for the composite TOE and not exposed at external interfaces:

- Hardware Special Function Register Access Control
- AES functionality of the Crypto Library (implemented by JCOP embedded SW instead)
- RSA functionality of the Crypto Library (implemented by JCOP embedded SW instead)
- Random Number Generator of the Crypto Library (implemented by JCOP embedded SW instead)
- Copy functionality of the Crypto Library (implemented by JCOP embedded SW instead)

1.3.2 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 [4] and has been adapted to Java Card specialties. The main actors are marked with bold letters.

Table 2. 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.

Phase	Name	Description
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-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	Personalisation	The **Personaliser** is responsible for • smartcard (including applet) personalization and final tests. Other 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.

The evaluation process is limited to phases 1 to 4.

The User Manual (AGD_OPE) for the applet developer and the Administrator Manual (AGD_PRE) are delivered in phase 1. The delivery of the hardware together with the other documentation is either in phase 3 or 4. The delivery comprises the following items:

Table 3. Delivery Items

Type	Name	Version	Date
Hardware	NXP J3A080 and J2A080 Secure Smart Card Controller Rev. 3		
	ROM Code (Mask ID)	51	
	Patch Code (Patch ID)	1	
Document	User Manual (AGD_OPE) for the applet developer	see Certification Report	see Certification Report
Document	Administrator Manual (AGD_PRE)	see Certification Report	see Certification Report
Document	HW Data Sheet [30]	see document	see document

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 not allowed. This means no 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 3.

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.

For details of the delivery process and possible delivery options please see Section 2 of the Guidance Manual of the hardware [29]. The delivery process for JCOP 2.4.1 R3 is exactly to same as the one defined for the hardware used by JCOP 2.4.1 R3.

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

1.3.3 Java Card Technology

For an overview on Java Card technology the reader is referred to Section 2 of the Java Card Protection Profile [1].

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 [1].

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 4. TOE Groups Overview

Group	Description	Scope

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. This group is not within the scope of evaluation.	--
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 was introduced in Java Card System 2.2.2. This group is not within 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 simultaneously selected or a single one can be selected more than once. This is a Java Card System 2.2.2 feature. 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. This is a Java Card System 2.2.2 feature. This group is not within the scope of evaluation.	--
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, a new feature introduced in Java Card System 2.2.2. It can also be used as a basis for any other application deletion requirements. This group is not within the scope of evaluation.	--
Secure carrier (CarG)	The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for	--

Group	Description	Scope
	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. This group is not within the scope of evaluation.	
Card manager (CMGRG)	The CMGRG contains the minimal requirements that allow defining a policy for controlling access to card content 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. This is a Java Card System 2.2.2 feature.	TOE

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

- Core (CoreG) (part of the TOE)
- Bytecode verification (BCVG) (part of the IT environment)

and is extended by the following groups:

- Smart card platform (SCPG) (part of the TOE)
- Card manager (CMGRG) (part of the TOE)
- External Memory (EMG) (part of the TOE)

Let us also remark 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 1.3.2. 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. The ST assumes that the loading of applications pre-issuance is made in a secure environment.

1.3.4 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.

1.3.5 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 J3A080 and J2A080, the certified hardware contains a native MIFARE application that belongs to the TOE. A TOE configured with the minor configuration option “MIFARE Emulation = A” does not provide an additional interface to the environment because the MIFARE application is logically disabled.

For J3A080 the minor configurations “MIFARE Emulation = B1” and “MIFARE Emulation = B4” implement the contactless MIFARE Classic OS and have access to 1KB or 4KB of EEPROM memory, respectively. The final product does not contain any other native applications according to JC PP. To completely securely separate the User OS and the MIFARE OS, which is enabled in minor configurations “MIFARE Emulation = B1” and “MIFARE Emulation = B4”, the smart card platform provides the so-called MIFARE firewall (see platform Security Targets [5]/**[22]**).

1.4 TOE Intended 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)

2.1 CC Conformance Claim

The ST and the composite TOE is conformant to Common Criteria version 3.1, because

- the TOE described in this ST is conformant to Common Criteria version 3.1 part 2 extended by FCS_RNG.1, FMT_LIM.1, FMT_LIM.2, FAU_SAS.1, and FPT_EMSEC.1;
- the TOE described in this ST is Common Criteria version 3.1 part 3 conformant, i.e. all assurance components are taken from part 3. The Evaluation Assurance Level is EAL 5 augmented by ALC_DVS.2 and AVA_VAN.5.

2.2 Statement of Compatibility concerning Composite Security Target

2.2.1 Separation of the Platform-TSF

This section describes the separation of relevant security functionality described in the ST of the smart card platform (Secured Crypto Library on the P5CD080V0B **[22]**) being used by this ST and others. The security functionality provided by the smart card platform is summarized in **[5]**, chapter 1.2.1. The following table confronts the relevant security functionality of the platform with those of the composite TOE defined in the present ST, in Table 5 the security functional requirements of the platform and of this composite ST are listed with the aim of separation of the platform functionality.

Table 5. Platform functionality

Platform functionality (Hardware and Crypto Library)	Usage by TOE	References / Remarks
AES • The AES algorithm is intended to provide encryption and decryption functionality. • The following modes of operation are supported for AES: ECB, CBC, CBC-MAC.	Hardware is used for AES (Advanced Encryption Standard) with key length of 128, 192, and 256 Bit for en-/decryption (CBC and ECB).	7.1.4
DES/3DES • The Single-DES algorithm can be used as a building	Hardware is used for 3DES (112 and 168 bit keys) for en-/decryption (CBC and ECB) and MAC generation and	7.1.4

Platform functionality (Hardware and Crypto Library)	Usage by TOE	References / Remarks
block, e.g. to implement a Retail-MAC. However, the Single-DES algorithm alone is not considered to be resistant against attacks with a high attack potential, therefore Single-DES alone must not be used for encryption. • The Triple-DES (3DES) algorithm is intended to provide encryption and decryption functionality. • The following modes of operation are supported for DES and Triple-DES: ECB, CBC, CBC-MAC.	verification (Retail-MAC and CBC-MAC). The BAC Accelerator uses the Crypto Library for 3DES (112 and 168 bit keys) for en-/decryption and MAC generation and verification (Retail-MAC).	
RSA • The RSA algorithm can be used for encryption and decryption as well as for signature generation and signature verification. • The RSA key generation can be used to generate RSA key pairs. • The RSA public key computation can be used to compute the public key that belongs to a given private key.	Hardware is used for RSA (1280 up to 2048 bits keys) for key generation, en-/decryption and signature generation and verification	7.1.4
ECC over GF(p) • The ECC over GF(p) algorithm can be used for signature generation and signature verification • The ECC over GF(p) key generation algorithm can be used to generate ECC over GF(p) key pairs. • The ECC Diffie-Hellman key exchange algorithm can be used to establish	Crypto Library is used for ECC over GF(p) for key length between 192 and 320 bits	7.1.4

Platform functionality (Hardware and Crypto Library)	Usage by TOE	References / Remarks
cryptographic keys. • The ECC over GF(p) point addition can be used for ECC point addition operations		
SHA • The SHA-1, SHA-224 and SHA-256 algorithms can be used for different purposes such as computing hash values in the course of digital signature creation or key derivation.	Crypto Library is used for SHA-1, SHA-224, and SHA-256 hash algorithm.	7.1.4
Resistance of cryptographic algorithms against side-channel attacks • The cryptographic algorithms (except SHA) are resistant against Side Channel Attacks, including Simple Power Analysis (SPA), Differential Power Analysis (DPA), Differential Fault Analysis (DFA) and timing attacks.	The TOE 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. All functions of SF.CryptoOperation except with SHA are resistant to side-channel attacks (e.g. timing attack, SPA, DPA, DFA, EMA, DEMA).	7.1.6
Random number generation • The TOE provides access to random numbers generated by a software (pseudo) random number generator and functions to perform the required test of the hardware (true) random number generator.	The TOE provides a hybrid random number generator that implements a deterministic RNG according to ANSI X9.31. It is seeded with random numbers from the physical RNG of the hardware.	7.1.4
Other security functionality • The TOE includes internal security measures for residual information protection. • The TOE provides a secure copy routine.	-	This functionality is used indirectly.

In the following table those SFRs of the platform are designated as “relevant” or “used by this composite ST”, which required functionality is also aimed or mentioned in the SFRs of this composite ST. The table also lists irrelevant Platform-SFRs not being used by the Composite-ST,

Table 6. Platform SFRs

Platform SFRs (Hardware and Crypto Library)	Usage by TOE, TOE-SFR	References / Remarks
Hardware SFRs		
FAU_SAS.1	SF.Hardware 5 and 7, FAU_SAS.1/SCP	7.1.9
FCS_RND.1	SF.Hardware 1, FCS_RNG.1	7.1.9
FDP_IFC.1	SF.Hardware 5 and 6, FDP_IFC.1/SCP	7.1.9
FDP_ITT.1	SF.Hardware 5 and 6, FDP_ITT.1/SCP	7.1.9
FMT_LIM.1	SF.Hardware 7, FCS_LIM.1	7.1.9
FMT_LIM.2	SF.Hardware 7, FCS_LIM.2	7.1.9
FPT_FLS.1	SF.Hardware 4 and 5, FPT_FLS.1/SCP	7.1.9
FPT_ITT.1	SF.Hardware 5 and 6, FPT_ITT.1/SCP	7.1.9
FPT_PHP.3	SF.Hardware 5, FPT_PHP.3/SCP	7.1.9
FPT_SEP.1	indirectly	Will be covered in ADV_ARC
FRU_FLT.2	SF.Hardware 4 and 5, FRU_FLT.2/SCP	7.1.9
FCS_COP.1[DES]	SF.Hardware 2 used for SF.CryptoOperation 1, FCS_COP.1/ DES	7.1.9 7.1.4
FCS_COP.1[AES]	SF.Hardware 3 used for SF.CryptoOperation 4, FCS_COP.1/ AES	7.1.9 7.1.4
FPT_SEP.1[CONF]	Indirectly	Will be covered in ADV_ARC
FDP_ACC.1[MEM]	SF.Hardware 8, FDP_ACC.1/SCP	7.1.9
FDP_ACC.1[SFR]	Irrelevant / Unused for TOE SFRs	-
FDP_ACF.1[MEM]	SF.Hardware 8, FDP_ACF.1/SCP	7.1.9
FDP_ACF.1[SFR]	Irrelevant / Unused for TOE SFRs	-
FMT_MSA.3[MEM]	SF.Hardware 8, FMT_MSA.3/SCP	7.1.9

Platform SFRs (Hardware and Crypto Library)	Usage by TOE, TOE-SFR	References / Remarks
FMT_MSA.3[SFR]	Irrelevant / Unused for TOE SFRs	-
FMT_MSA.1[MEM]	Irrelevant / Unused for TOE SFRs	-
FMT_MSA.1[SFR]	Irrelevant / Unused for TOE SFRs	-
FMT_SMF.1	Irrelevant / Unused for TOE SFRs	-
SFRs additionally defined in the smart card platform (Crypto Library) Security Target		
FDP_IFC.1	SF.CryptoLib 14 – Hardware functionality in extended by Crypto Library and used for SF.CryptoLib 2, 6, 7 and 9, FDP_IFC.1/SCP	7.1.10
FDP_ITT.1	SF.CryptoLib 14 – Hardware functionality in extended by Crypto Library and used for SF.CryptoLib 2, 6, 7 and 9, FDP_ITT.1/SCP	7.1.10
FPT_ITT.1	SF.CryptoLib 14 – Hardware functionality in extended by Crypto Library and used for SF.CryptoLib 2, 6, 7 and 9, FPT_ITT.1/SCP	7.1.10
FPT_FLS.1	SF.CryptoLib 14 - Failure with preservation of secure state used for SF.CryptoLib 2, 6, 7 and 9, FPT_FLS.1	7.1.10
FCS_COP.1[SW-AES]	Irrelevant / Unused for TOE SFRs	-
FCS_COP.1[SW-DES]	SF.CryptoLib 2 used for SF.CryptoOperation 12 (BAC Accelerator), FCS_COP.1/TDES_MRTD and FCS_COP.1/MAC_MRTD	7.1.10 7.1.4
FCS_COP.1 [RSA_encrypt]	Irrelevant / Unused for TOE SFRs	-
FCS_COP.1 [RSA_public]	Irrelevant / Unused for TOE SFRs	-
FCS_COP.1[RSA_sign]	Irrelevant / Unused for TOE SFRs	-
FCS_COP.1 [ECC_GF_p]	SF.CryptoLib 6 used for SF.CryptoOperation 9, FCS_COP.1/ECSignature	7.1.10 7.1.4
FCS_COP.1 [ECC_ADD]	SF.CryptoLib 7 used for SF.CryptoOperation 14, FCS_COP.1/ECAdd	7.1.10 7.1.4
FCS_COP.1 [ECC_DHKE]	SF.CryptoLib 7 used for SF.CryptoOperation 13, FCS_COP.1/DHKeyExchange	7.1.10 7.1.4

Platform SFRs (Hardware and Crypto Library)	Usage by TOE, TOE-SFR	References / Remarks
FCS_COP.1[SHA]	SF.CryptoLib 10 used for SF.CryptoOperation 7, 10, and 11, FCS_COP.1/ SHA-1, FCS_COP.1/SHA-224, FCS_COP.1/SHA-256	7.1.10 7.1.4
FCS_CKM.1[RSA]	Irrelevant / Unused for TOE SFRs	
FCS_CKM.1 [ECC_GF_p]	SF.CryptoLib 9 used for SF.CryptoKey 10, FCS_CKM.1	7.1.10 7.1.3
FDP_RIP.1	SF.CryptoLib 12 used for for SF.CryptoLib 2, 6, 7 and 9, FDP_RIP.1/SCP	7.1.10
FDP_ITT.1[COPY]	Irrelevant / Unused for TOE SFRs	-
FPT_ITT.1[COPY]	Irrelevant / Unused for TOE SFRs	-
FCS_RNG.1[DET]	Irrelevant / Unused for TOE SFRs	-
FPT_TST.2	Irrelevant / Unused for TOE SFRs	-

The assurance classes are covered as follows:

The smart card platform is EAL5, augmented with AVA_VLA.4 according Common Criteria 2.3. The composite TOE is EAL 5, augmented by ALC_DVS.2 and AVA_VAN.5 according Common Criteria 3.1.

The component ALC_DVS.2 of Common Criteria 2.3 and 3.1 are the equivalent.

AVA_VAN.5 is covered by AVA_MSU.3 and AVA_VLA.4.

2.2.2 Compatibility between the Composite Security Target and the Platform Security Target

The following mappings in Table 7, Table 8, Table 10 , Table 10, and Table 11 demonstrates the compatibility between the Composite Security Target (the document at hand) and the smart card platform Security Target **[22]** regarding threats, assumptions, organisational security policies and objectives.

There is no conflict between threats of the Composite Security Target and the smart card platform Security Target.

Table 7. Platform Threats

Platform threats (Hardware and Crypto Library)	Platform Objective	Pendant by TOE	Remarks
T.Leak-Inherent	O.Leak-Inherent	T.LEAKAGE	See 3.3.1.4
T.Phys-Probing	O.Phys-Probing O.Phys-	T.PHYSICAL	See 3.3.1.4

Platform threats (Hardware and Crypto Library)	Platform Objective	Pendant by TOE	Remarks
T.Phys-Manipulation	Manipulation		
T.Malfunction T.Leak-Forced	O.Malfunction O.Leak-Forced	T.FAULT	See 3.3.1.4
T.Abuse-Func	O.Abuse-Func	T.OS_DECEIVE T.OS_OPERATE	See 3.3.1.3
T.RND	O.RND	T.RND	See 3.3.1.5
-		T.SID.1 T.SID.2 T.CONFID-JCS-CODE T.CONFID-APPLI-DATA T.CONFID-JCS-DATA T.INTEG-APPLI-CODE T.INTEG-JCS-CODE T.INTEG-APPLI-DATA T.INTEG-JCS-DATA T.EXE-CODE.1 T.EXE-CODE.2 T.RESOURCES T.ACCESS_DATA	Additional Threats on OS

There is no conflict between OSPs of the Composite Security Target and the smart card platform Security Target.

Table 8. Platform OSPs

Platform OSPs (Hardware and Crypto Library)	Platform Objective	Applicable threats by TOE	Remarks
P.Add-Components	O.HW_AES O.HW_DES3 O.MF_FW O.Leak-Inherent O.Phys-Probing O.Malfunction O.Phys-Manipulation O.Leak-Forced O.MEM_ACCESS O.SFR_ACCESS	T.LEAKAGE T.PHYSICAL T.FAULT T.ACCESS_DATA T.OS_DECEIVE	The OSP is not contradictory to the threats of the Composite-ST because it refers to threats that are consistent to the OSP.
P.Add-Func	O.AES O.DES3 O.RSA O.RSA_PubKey O.RSA_KeyGen O.ECC O.ECC_DHKA	T.LEAKAGE T.PHYSICAL T.FAULT	The OSP is not contradictory to the threats of the Composite-ST because it refers to threats that are

Platform OSPs (Hardware and Crypto Library)	Platform Objective	Applicable threats by TOE	Remarks
	O.ECC_KeyGen O.SHA O.RND O.REUSE O.COPY O.MEM_ACCESS O.Leak-Inherent O.Phys-Probing O.Malfunction O.Phys-Manipulation O.Leak-Forced		consistent to the OSP. The key size used by the TOE for ECC operations is a subset of possible values in the crypto library.
P.Process-TOE	OE.Process-TOE O.Identification	T.OS_DECEIVE	The OSP is not contradictory to the threats of the Composite-ST part of the composite ST because it refers to threats that are to the OSP.
		T.OS_OPERATE T.SID.1 T.SID.2 T.CONFID-JCS-CODE T.CONFID-APPLI-DATA T.CONFID-JCS-DATA T.INTEG-APPLI-CODE T.INTEG-JCS-CODE T.INTEG-APPLI-DATA T.INTEG-JCS-DATA T.EXE-CODE.1 T.EXE-CODE.2 T.RESOURCES	There is no contradiction to the platform since there are no applicable relevant platform OSPs.

There is no conflict between security assumptions of the Composite Security Target and the smart card platform Security Target.

Table 9. Platform Assumptions

Platform assumptions (Hardware and Crypto Library)	Platform Objectives	Pendant by TOE	Remarks
A.Process-Card	OE.Process-Card	-	This assumption is not about the *operational* environment as required by CC 3.1.
A.Plat-Appl	OE.Plat-Appl	-	This assumption is not about the *operational* environment as required by CC 3.1
A.Resp-Appl	OE.Resp-Appl	A.USE_KEYS	Security relevant User Data (especially cryptographic keys) are treated by the Smartcard Embedded Software as required by the security needs of the specific application context. Additionally, it is assumed that keys in the environment of the composite TOE are protected for confidentiality and integrity. See 3.5.1.1.
A.Check-Init	OE.Check-Init	-	The smartcard software checks initialization data as described in the objective O.OPERATE
A.Key-Function	OE.Plat-Appl OE.Resp-Appl	A.USE_KEYS	See A.Platt-Appl and A.Resp-Appl
-	-	A.USE_DIAG A.NATIVE A.NO-DELETION A.NO-INSTALL A.VERIFICATION	Additional assumptions of OS

There is no conflict between security objectives of the Composite Security Target and the smart card platform Security Target.

Table 10. Platform Objectives

Platform objectives (Hardware and Crypto Library)	Platform SFRs	Pendant by TOE	Remarks
Hardware Objectives			
O.Leak-Inherent	FDP_ITT.1 FPT_ITT.1 FDP_IFC.1	O.SIDE_CHANNEL	See 4.1.1.1
O.Phys-Probing O.Phys-Manipulation	FPT_PHP.3	O.PHYSICAL O.SCP.IC	See 4.1.1.2 and 4.1.2.5
O.Malfunction	FRU_FLT.2	O.FAULT_PROTECT	For Fault

Platform objectives (Hardware and Crypto Library)	Platform SFRs	Pendant by TOE	Remarks
	FPT_FLS.1 FPT_SEP.1	O.SCP.RECOVERY O.SCP.SUPPORT	protection see 4.1.1.1. The SCP allows the TOE to eventually complete the interrupted operation successfully after startup, see 4.1.2.5.
O.Leak-Forced	FDP_ITT.1 FPT_ITT.1 FDP_IFC.1 FPT_PHP.3	O.FAULT_PROTECT O.PHYSICAL	See 4.1.1.1 and 4.1.1.2
O.Abuse-Func	FMT_LIM.1 FMT_LIM.2 and see O.Leak-Inherent, O.Phys-Probing, O.Malfunction, O.Phys-Manipulation, O.Leak-Forced	O.PROTECT_DATA O.OS_DECEIVE	See 4.1.1.1
O.Identification	FAU_SAS.1	O.IDENTIFICATION	Identification is provided by platform, see 4.1.1.2
O.RND	FCS_RND.1 and see O.Leak-Inherent, O.Phys-Probing, O.Malfunction, O.Phys-Manipulation, O.Leak-Forced	O.RND	Random number generation is supported by platform, see 4.1.1.3
O.HW_DES3 O.HW_AES	FCS_COP.1[DES] FCS_COP.1[AES]	O.CIPHER (parts)	See 4.1.2.3
O.MF_FW	FDP_ACC.1[MEM] FDP_ACF.1[MEM] FMT_MSA.3[MEM]	O.MF_FW	See 4.1.1.2
O.MEM_ACCESS O.SFR_ACCESS	FDP_ACC.1[MEM] FDP_ACF.1[MEM] FMT_MSA.3[MEM] FMT_MSA.1[MEM] FMT_MSA.1[SFR] FMT_SMF.1 FDP_ACC.1[SFR] FDP_ACF.1[SFR] FMT_MSA.3[SFR] FMT_MSA.1[SFR]	Unused for TOE	Not contradicting

Platform objectives (Hardware and Crypto Library)	Platform SFRs	Pendant by TOE	Remarks
	FMT_SMF.1		
O.CONFIG	FPT_SEP.1[CONF]	-	Will be covered in ADV_ARC
SFRs additionally defined in the smart card platform (Crypto Library) Security Target			
O.AES O.RSA O.RSA_PubKey O.RSA_KeyGen	*FCS_COP.1[SW-AES]* *FCS_COP.1[RSA_encrypt]* *FCS_COP.1[RSA_sign]* *FCS_COP.1[RSA_public]* *FCS_CKM.1[RSA]* *FDP_IFC.1* *FDP_ITT.1* *FPT_ITT.1* *FPT_FLS.1* *FRU_FLT.2*	*Unused for TOE*	Not contradicting
O.DES3 O.ECC O.ECC_DHKE O.ECC_KeyGen O.SHA	FCS_COP.1[SW-DES] FCS_COP.1[ECC_GF_p] FCS_COP.1[ECC_ADD] FCS_COP.1[ECC_DHKE] FCS_CKM.1[ECC_GF_p] FCS_COP.1[SHA] FDP_IFC.1 FDP_ITT.1 FPT_ITT.1 FPT_FLS.1 FRU_FLT.2	O.CIPHER (parts)	See 4.1.2.3
O.COPY	FDP_ITT.1[COPY] FPT_ITT.1[COPY]	-	Functionality is not used
O.REUSE	FDP_RIP.1	O.REALLOCATION	Memory reallocation is supported by platform
O.RND	FCS_RND.2 FPT_TST.2	Unused for TOE	Not contradicting
-	-	O.SID O.OPERATE O.RESOURCES O.FIREWALL O.SHRD_VAR_INTEG O.SHRD_VAR_CONFID	Additional objective of OS

Platform objectives (Hardware and Crypto Library)	Platform SFRs	Pendant by TOE	Remarks
		O.ALARM O.TRANSACTION O.PIN-MNGT O.KEY-MNGT O.CARD-MANAGEMENT	

There is no conflict between security objectives for the environment of the Composite Security Target and the smart card platform Security Target.

Table 11. Platform Objectives Environment

Platform objectives (Hardware and Crypto Library)	Pendant by TOE	Remarks
OE.Plat-Appl	-	This assumption is not about the *operational* environment as required by CC 3.1
OE.Resp-Appl	-	This assumption is not about the *operational* environment as required by CC 3.1
OE.Process-TOE	-	This assumption is not about the *operational* environment as required by CC 3.1
OE.Process-Card	-	This assumption is not about the *operational* environment as required by CC 3.1
OE.Check-Init	O.OPERATE	The smartcard software checks initialization data as described in the objectives
-	OE.USE_DIAG OE.USE_KEYS OE.NATIVE OE.NO-DELETION OE.NO-INSTALL OE.VERIFICATION	Additional objectives of OS

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. To describe the threats the assets are defined in the next section. The threats are described in section 3.3. The last section 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 [4] and supplemented by the description of [1].

3.2 Assets

In this section assets are divided in primary and secondary assets. The primary assets User Data and TSF Data are further refined, and the definition of section 3.2 of Java Card System - Minimal Configuration Protection Profile [1] is taken.

The TOE objective is to protect assets, the primary assets, during usage phase. In order to protect these primary assets, information and tools used for the development and manufacturing of the Smart Card, need to be protected. These information and tools are called secondary assets.

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.

Java Card specific (primary) assets defined in [1] to be protected by the TOE are listed below. They are grouped according to whether it is data created by and for the user (User data) or data created by and for the TOE (TSF data). For each asset it is specified the kind of dangers that weighs on it.

3.2.1 User Data

D.APP_CODE	The code of the *applet*s 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 *applet*s.

	To be protected from unauthorized disclosure and modification.

3.2.2 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 *AID*s used to identify the installed *applet*s, 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.JCS_KEYs	Cryptographic keys used when loading a file into the card. 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.

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 [1]. Only threats on TOE information during phase 7 are considered. They are summarized in the following table:

Table 12. Threats

Name	Source	Refined?
T.ACCESS_DAT	-	-

Name	Source	Refined?
A		
T.OS_OPERATE	-	-
T.OS_DECEIVE	-	-
T.LEAKAGE	-	-
T.FAULT	-	-
T.RND	[4]	no
T.PHYSICAL	[1]	yes[3]
T.CONFID-JCS-CODE	[1]	no
T.CONFID-APPLI-DATA	[1]	no
T.CONFID-JCS-DATA	[1]	no
T.INTEG-APPLI-CODE	[1]	no
T.INTEG-JCS-CODE	[1]	no
T.INTEG-APPLI-DATA	[1]	no
T.INTEG-JCS-DATA	[1]	no
T.SID.1	[1]	no
T.SID.2	[1]	no
T.EXE-CODE.1	[1]	no
T.EXE-CODE.2	[1]	no
T.RESOURCES	[1]	no

3.3.1 Threats not contained in [1]

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] Refinement to cover additional aspects of O.SCP.IC not contained in [1].

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

- 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.ACCESS_DATA	Unauthorized access to sensitive information stored in memories in order to disclose or to corrupt the TOE data (TSF and user data).

	This includes any consequences of bad or incorrect user authentication by the TOE. Several software attack methods may be used here, as: • Brute force data space search attacks, • Administrator or user authentication failures information • Monitoring TOE inputs/outputs, • Replay attacks, • Cryptographic attacks, Regarding direct attacks on the Native Operating System, there are series of attack paths that address logical probing of the TOE. These are brute force data search, replay attack, insertion of faults, and invalid input.
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.OS_DECEIVE	Modification of the expected TOE configuration by • unauthorized loading of code,

- unauthorized execution of code
- unauthorized modification of code behavior

Complementary note: Any software manipulations on TOE application data (User or TSF data) by interaction with other program or security features that may not be used after one given life cycle state can lead to unsecured state.

The attacker needs to know specific information about the TOE implementation. Loading of code in EEPROM (patch or filter) is considered here as authorized, if this code is part of the evaluation and if loading operation is performed by an authorized administrator in the defined environment.

3.3.1.4 Environment Threats on the complete TOE

Regarding physical point of view, the TOE is exposed to different types of influences or interactions with its outer world. One can take advantage of the leakage from the TOE to disclose assets or derived data.

T.LEAKAGE: An attacker may exploit information which is leaked from the TOE during usage of the Smart Card in order to disclose the confidential primary assets.

This attack is non-invasive and requires no direct physical contact with the Smart Card Internals. Leakage may occur through emanations, variations in power consumption, I/O characteristics, clock frequency, or by changes in processing time requirements. One example is the Differential Power Analysis (DPA).

Another security concern is to take advantage of the susceptibility of the integrated circuit to put the TOE in an unsecured state.

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

Complementary note: A composition attack paths can take advantage of any susceptibility of the whole product (ES and IC):

- environmental variations (temperature, power supply, clock frequency) or exposition to strenuous particles (light and electrical waves)

The attacker uses the available interfaces of the TOE

3.3.1.5 Threat on Random Numbers

The following threat was taken over from [4]:

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 [1]

The following threats specific for the Java Card functionality were taken from [1].

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 using pure invasive, physical (opposed to logical) attacks on the hardware part of the TOE.

These attacks are performed using physical probing or physical manipulation of the hardware (reverse engineering, manipulation of memory cells, manipulation of hardware security parts) and include IC failure analysis, electrical probing, unexpected tearing, and DP analysis. 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 #.SCP.7, and all aspects related to confidentiality and integrity of code and data.

Note: This threat from [1] was refined to cover additional aspects of O.SCP.IC not contained in [1].

3.3.2.1 Confidentiality

T.CONFID-JCS-CODE — The attacker executes an application without authorization to disclose the *Java Card System* code. See #.CONFID-JCS-CODE (p. 39) for details.

Directly threatened asset(s): D.JCS_CODE.

T.CONFID-APPLI-DATA — The attacker executes an application without authorization to disclose data belonging to another application. See #.CONFID-APPLI-DATA (p. 39) for details.

Directly threatened asset(s): D.APP_C_DATA, D.PIN and D.APP_KEYs.

T.CONFID-JCS-DATA	The attacker executes an application without authorization to disclose data belonging to the *Java Card System*. See #.CONFID-JCS-DATA (p. 39) 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. 39) for details. Directly threatened asset(s): D.APP_CODE
T.INTEG-JCS-CODE	The attacker executes an application to alter (part of) the *Java Card System* code. See #.INTEG-JCS-CODE (p. 39) for details. Directly threatened asset(s): D.JCS_CODE.
T.INTEG-APPLI-DATA	The attacker executes an application to alter (part of) another application's data. See #.INTEG-APPLI-DATA (p. 39) for details. Directly threatened asset(s): D.APP_I_DATA, D.PIN and D.APP_KEYs.
T.INTEG-JCS-DATA	The attacker executes an application to alter (part of) *Java Card System* or API data. See #.INTEG-JCS-DATA (p. 40) 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. 42) for details. Directly threatened asset(s): D.SEC_DATA (other assets may be jeopardized should this attack succeed, for instance, if the identity of the *JCRE* is usurped), D.PIN, D.APP_KEYs and D.JCS_KEYs
T.SID.2	The attacker modifies the identity of the privileged roles. See #.SID (p. 42) 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. 40) and #.EXE-APPLI-CODE (p. 40) 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. 40) and #.EXE-APPLI-CODE (p. 40) for details. Directly threatened asset(s): D.APP_CODE.

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. Directly threatened asset(s): D.JCS_DATA.

3.4 Organisational security policies (OSPs)

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

3.4.1 Organizational Security Policies from [1]

There is no Organizational Security Policiy (OSP) in [1] for the minimal configuration.

3.4.2 Additional Organizational Security Policies

The IC Developer / Manufacturer must apply the policy “Protection during TOE Development and Production (OSP.PROCESS-TOE)” as specified below.

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

3.5 Assumptions

This section introduces the assumptions made on the environment of the TOE. They are summarized in the following table together with the life-cycle phase they apply to:

Table 13. Assumptions

Name	Source	Refined?	Life-Cycle
A.USE_DIAG	-	-	phase 7
A.USE_KEYS	-	-	phase 7
A.NATIVE	[1]	no	phases 1-6
A.NO-DELETION	[1]	no	phase 7

Name	Source	Refined?	Life-Cycle
A.NO-INSTALL	[1]	no	phase 7
A.VERIFICATION	[1]	no	phases 1-6

3.5.1 Assumptions not contained in [1]

3.5.1.1 Assumption on Phase 7

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.

Application 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.

3.5.2 Assumptions from [1]

A.NATIVE — Those parts of the APIs written in native code as well as any pre-issuance native application on the card are assumed to be conformant with the TOE so as to ensure that security policies and objectives described herein are not violated. See #.NATIVE (p. 40) for details

Remark: A.NATIVE is related to all phases except phase 7 as in the usage phase, no more native code can be loaded onto the card

A.NO-DELETION — No deletion of installed applets (or packages) is possible.

Remark: A.NO-DELETION is related to phase 7.

A.NO-INSTALL — There is no post-issuance installation of applets. Installation of applets is secure and occurs only in a controlled environment in the pre-issuance phase. See #.INSTALL (p. 42) for details.

Remark: This assumption is related to phase 7.

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.

Remark: A.VERIFICATION is related to phases 1-6 since in the present case, bytecode verification is performed before loading.

3.6 Security Aspects

This section is partly taken from [1].

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 O.FIREWALL.

The following sections present several security aspects from [1] 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 ([11],§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 ([11],§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 instances[4], and isolation of their data and code between *package*s (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 file*s and the assurance that the export files used to check the *CAP file*

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

correspond to those that will be present on the card when loading occurs.

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 [9], [12]). **The actual set of checks performed by the verifier is implementation-dependent, but shall at least enforce all the "must clauses" imposed in [9] on the bytecodes and the correctness of the CAP files' format.**

As most of the actual *JCVM*s 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 [9], §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 *package*s 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 *applet*s. **(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 *applet*s. 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 *applet*s. **(3)** The procedure of loading and installing a package shall ensure its integrity and authenticity.
Note:	In the present case, applet installation takes place in a secure environment prior to card issuing.
#.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 *applet*s (or *package*s) 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 *applet*s. 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 *package*s can be declared to be undeletable (for instance, the Java Card API *package*s), or the dependency between installed *package*s may forbid the deletion (like a *package* using super classes or super interfaces declared in another package).

Note: In the present case, deletion of applets is not allowed.

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 *package*s 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 (ASE_OBJ)

4.1 Security objectives for the TOE

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

Table 14. Security Objectives for the TOE

Name	Source	Refined?
O.PROTECT_DATA	-	-
O.SIDE_CHANNEL	-	-
O.OS_DECEIVE	-	-
O.FAULT_PROTECT	-	-
O.PHYSICAL	-	-
O.IDENTIFICATION	[4]	no
O.RND	[4]	no
O.SID	[1]	no
O.MF_FW	[5]	no
O.OPERATE	[1]	yes[5]
O.RESOURCES	[1]	no
O.FIREWALL	[1]	no
O.REALLOCATION	[1]	no
O.SHRD_VAR_CONFID	[1]	no
O.SHRD_VAR_INTEG	[1]	no
O.ALARM	[1]	no
O.TRANSACTION	[1]	no
O.CIPHER	[1]	no
O.PIN-MNGT	[1]	no
O.KEY-MNGT	[1]	no
O.CARD-MANAGEMENT	[1]	no (*)

[5] Refinement to cover additional aspects of threat T.OS.Operate not contained in [1].

Name	Source	Refined?
O.SCP.RECOVERY	[1]	no (*)
O.SCP.SUPPORT	[1]	no (*)
O.SCP.IC	[1]	no (*)

(*) These Security Objectives for the environment of [1] are Security Objectives for the TOE in the present evaluation. Therefore, the label changed (O.XYZ instead of OE.XYZ) but not the content (no refinement).

4.1.1 Security Objectives for the TOE not contained in [1]

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
- Protection of the TOE and associated documentation and environment during development and production phases.

The TOE shall use state of the art technology to achieve the following IT security objectives, and for that purpose, when IC physical security features are used, the specification of those IC physical security features shall be respected.

When IC physical security features are not used, the Security Objectives shall be achieved in other ways.

4.1.1.1 Security Objectives for the complete TOE

O.PROTECT_DATA — The TOE shall ensure that sensitive information stored in memories is protected against unauthorized disclosure and any corruption or unauthorized modification.

Moreover, the TOE shall ensure that sensitive information stored in memories is protected against unauthorized access.

The TOE has to provide appropriate security mechanisms to avoid fraudulent access to any sensitive data, such as passwords, cryptographic keys or authentication data.

This is obvious for secret information, but also applies to access controlled information sensitive information means here, any primary assets that can be refined by the TOE developer in the Security Target and need to be protected.

O.SIDE_CHANNEL — The TOE must provide protection against disclosure of primary assets including confidential data (User Data or TSF data) stored and/or processed in the Smart Card IC:

- by measurement and analysis of the shape and amplitude

- by measurement and analysis of the time between events found by measuring signals (for example on the power, clock, or I/O lines).

Especially, the NOS must be designed to avoid interpretations of signals extracted, intentionally or not, from the hardware part of the TOE (for instance, Power Supply, Electro Magnetic emissions).

O.OS_DECEIVE	The TOE must guarantee that only secure values are used for its management and operations, especially system flags or cryptographic assets. Moreover, the integrity of the whole TOE including the NOS must be guaranteed to prevent • Disclosing/bypassing of the NOS mechanisms, or • Modifying the expected NOS behavior (for instance, unauthorized code patch, or rewriting).
O.FAULT_PROTECT	The TOE must ensure its correct operation even outside the normal operating conditions where reliability and secure operation has not been proven or tested. This is to prevent errors. The environmental conditions may include voltage, clock frequency, temperature, or external energy fields that can be applied on all interfaces of the TOE (physical or electrical).
O.PHYSICAL	The TOE hardware provides the following protection against physical manipulation of the IC, and prevent • Reverse-engineering (understanding the design and its properties and functions), • Physical access to the IC active surface (probing) allowing unauthorized memory content disclosure, • Manipulation of the hardware security parts (e.g. sensors, cryptographic engine or RNG), • Manipulation of the IC, including the embedded NOS and its application data (e.g. lock and life cycle status, authentication flags, etc.).

4.1.1.2 Additional Security Objectives for the IC

The hardware participates to objectives of the complete TOE and has to fulfill a specific objective. This objective must be the result of the state of the art of Integrated Circuit security mechanisms. The precise statement of the corresponding requirements for this objective is made in the ST of the hardware platform, which is compliant to the:

- Smart Card IC Platform Protection Profile [4].

O.IDENTIFICATION	The TOE must provide means to store Initialisation Data and Pre-personalisation Data in its non-volatile memory. The Initialisation Data (or parts of them) are used for TOE identification.

O.MF_FW	The TOE shall provide separation between the “MIFARE Operating System” IC Dedicated Support Software and the Smartcard Embedded Software. The separation shall comprise software execution and data access.

4.1.1.3 Security Objective concerning Random Numbers

The following security objective was taken over from [4]:

O.RND	Random Numbers The TOE will ensure the cryptographic quality of random number generation. For instance random numbers shall not be predictable and shall have a 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 [1]

4.1.2.1 Identification

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

4.1.2.2 Execution

O.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 43) for details.
O.RESOURCES	The TOE shall control the availability of resources for the applications. See #.RESOURCES (p 43) for details.
O.FIREWALL	The TOE shall ensure controlled sharing of data containers owned by *applets* of different *package*s, and between *applets* and the TSFs. See #.FIREWALL (p 40) for details.
O.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. *Application note: To be made unavailable means to be physically erased with a default value. Except for local variables that do not correspond to method parameters, the default values to be used are specified in Java Card Virtual Machine Specification [9].*
O.SHRD_VAR_CONFID	The TOE shall ensure that any data container that is shared by all applications is always cleaned after the execution of an application. Examples of such shared containers are the APDU buffer, the byte array used for the invocation of the process method of the selected applet, or any public global variable exported by the API.
O.SHRD_VAR_INTEG	The TOE shall ensure that only the currently selected application may grant write access to a data memory area that is shared by all applications, like the APDU buffer, the byte array used for the invocation of the process method of the

selected applet, or any public global variable exported by the API. Even though the memory area is shared by all applications, the TOE shall restrict the possibility of getting a reference to such memory area to the application that has been selected for execution. The selected application may decide to temporarily hand over the reference to other applications at its own risk, but the TOE shall prevent those applications from storing the reference as part of their persistent states.

4.1.2.3 Services

O.ALARM The TOE shall provide appropriate feedback information upon detection of a potential security violation. See #.ALARM (p. 43) for details.

O.TRANSACTION The TOE must provide a means to execute a set of operations atomically. See #.TRANSACTION (p. 44) for details.

O.CIPHER The TOE shall provide a means to cipher sensitive data for applications in a secure way. In particular, the TOE must support cryptographic algorithms consistent with cryptographic usage policies and standards. See #.CIPHER (p. 43) for details.

O.PIN-MNGT The TOE shall provide a means to securely manage PIN objects. See #.PIN-MNGT (p. 44) for details.

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

O.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. 44).

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

Note: For this Java Card such libraries do not exist. All necessary functionality is implemented by the TOE.

4.1.2.4 Card Management

The TOE Security Objective for the card manager is a Security Objective for the environment in [1]. In the present case the card manager belongs to the TOE and the corresponding Security Objective is listed here.

O.CARD-MANAGEMENT The card manager shall control the access to card management functions such as the installation, update or

deletion of *applet*s. It shall also implement the card issuer's policy on the card.

4.1.2.5 Smart Card Platform

These TOE Security Objectives for the smart card platform are Security Objectives for the environment in [1]. In the present case the certified smart card platform belongs to the TOE and the corresponding Security Objectives are listed here.

O.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).

O.SCP.SUPPORT — The *SCP* shall provide functionalities that support the well-functioning of the TSFs of the TOE (avoiding they are bypassed or altered) and by controlling the access to information proper of the TSFs. In addition, the smart card platform should also provide basic services which are required by the runtime environment to implement security mechanisms such as atomic transactions, management of persistent and transient objects and cryptographic functions. These mechanisms are likely to be used by security functions implementing the security requirements defined for the TOE. See #.SCP.2-5 (p.44).

O.*SCP*.IC — The *SCP* shall possess IC security features. See #.SCP.7 (p.44).

4.2 Security objectives for the operational environment

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

Table 15. Security Objectives for the operational environment

Name	Source	Refined?
OE.USE_DIAG	-	-
OE.USE_KEYS	-	-
OE.NATIVE	[1]	no
OE.NO-DELETION	[1]	no
OE.NO-INSTALL	[1]	no
OE.VERIFICATION	[1]	no

4.2.1 Security Objectives for the operational environment not contained in [1]

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. *Application note: 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)*

4.2.2 Security Objectives for the operational environment from [1]

OE.NATIVE	Those parts of the APIs written in native code as well as any pre-issuance native application on the card shall be conformant with the TOE so as to ensure that security policies and objectives described herein are not violated. See #.NATIVE (p.40) for details.
Note:	The Security Objectives from [1] for the environment *OE.SCP.RECOVERY*, *OE.SCP.SUPPORT*, and *O.SCP.IC* are listed as TOE security objectives for the TOE in section 4.1.2.5 as the smart card platform belong to the TOE for this evaluation.
Note:	The Security Objective from [1] for the environment *OE.CARD-MANAGEMENT* is listed as TOE security objective for the TOE in section 4.1.2.4 as the Card Manager belongs to the TOE for this evaluation.
OE.NO-DELETION	No installed *applet*s (or *package*s) shall be deleted from the card.

OE.NO-INSTALL There is no post-issuance installation of *applet*s. Installation of *applet*s is secure and shall occur only in a controlled environment in the pre-issuance phase.

The objectives *OE.NO-INSTALL* and *OE.NO-DELETION* have been included so as to describe procedures that shall contribute to ensure that the TOE will be used in a secure manner. Moreover, they have been defined in accordance with the environmental assumptions they uphold (actually, they are just a reformulation of the corresponding assumptions). The *NO-DELETION* and *NO-INSTALL* (assumptions and objectives) constitute the explicit statement that the Minimal configuration corresponds to that of a closed card (no code can be loaded or deleted once the card has been issued). It is not evident that these objectives should be carried out by using IT means.

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.40) for details.

4.3 Relation between security objectives and the security problem definition

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

	O.PROTECT_DATA	O.OS_DECEIVE	O.SIDE_CHANNEL	O.FAULT_PROTECT	O.IDENTIFICATION	O.PHYSICAL	O.CARD-MANAGEMENT	O.SHRD_VAR_INTEG	O.SHRD_VAR_CONFID	O.FIREWALL	O.OPERATE	O.ALARM	O.RESOURCES	O.REALLOCATION	O.SID	O.MF_FW	O.SCP.IC	O.SCP.RECOVERY	O.SCP.SUPPORT	O.CIPHER	O.PIN-MNGT	O.KEY-MNGT	O.TRANSACTION	O.RND
T.ACCESS_DATA	x																							
T.OS_OPERATE	x										x					x								
T.OS_DECEIVE		x																						
T.LEAKAGE			x																					

	O.PROTECT_DATA	O.OS_DECEIVE	O.SIDE_CHANNEL	O.FAULT_PROTECT	O.IDENTIFICATION	O.PHYSICAL	O.CARD-MANAGEMENT	O.SHRD_VAR_INTEG	O.SHRD_VAR_CONFID	O.FIREWALL	O.OPERATE	O.ALARM	O.RESOURCES	O.REALLOCATION	O.SID	O.MF_FW	O.SCP.IC	O.SCP.RECOVERY	O.SCP.SUPPORT	O.CIPHER	O.PIN-MNGT	O.KEY-MNGT	O.TRANSACTION	O.RND
T.FAULT				x																				
T.PHYSICAL						x											x							
T.CONFID-JCS-DATA T.INTEG-JCS-DATA							x			x	x	x			x			x	x					
T.CONFID-APPLI-DATA							x		x	x	x	x	x		x			x	x	x	x	x	x	
T.INTEG-APPLI-DATA							x	x		x	x	x	x		x			x	x	x	x	x	x	
T.SID.1							x			x					x									
T.SID.2										x	x				x			x	x					
T.RESOURCES											x			x				x	x					
T.RND																								x
OSP.PROCESS-TOE					x																			

Table 17. Assignment: threats / assumptions / OSP – security objectives for the environment

	OE.NO-INSTALL	OE.NO-DELETION	OE.VERIFICATION	OE.USE_DIAG	OE.USE_KEY	OE.NATIVE

	OE.NO-INSTALL	OE.NO-DELETION	OE.VERIFICATION	OE.USE_DIAG	OE.USE_KEY	OE.NATIVE
T.CONFID-JCS-CODE T.INTEG-APPLI-CODE T.INTEG-JCS-CODE			x			
T.CONFID-JCS-DATA T.INTEG-JCS-DATA			x			
T.CONFID-APPLI-DATA			x			
T.INTEG-APPLI-DATA			x			
T.EXE-CODE.1			x			
T.EXE-CODE.2			x			
A.USE_DIAG				x		
A.USE_KEY					x	
A.NATIVE						x
A.NO-DELETION		x				
A.NO-INSTALL	x					
A.VERIFICATION			x			

Parts of the tables were reproduced from the information contained in [4] section 7.1, and [1] section 6.1.1. The justifications given in these sections were slightly adapted and taken over in this ST for completeness reasons.

The following three points must be taken into account when comparing the justification from [4] and [1] with the justifications given below:

- O.OPERATE was refined to cover additional aspects of threat T.OS.Operate not contained in [1].
- O.CARD-MANAGEMENT, O.SCP.RECOVERY, O.SCP.SUPPORT, and O.SCP.IC are security objectives for the environment in [1]. They are independent of the other objectives. So the justifications from [1] for OE.CARD-MANAGEMENT, OE.SCP.RECOVERY, OE.SCP.SUPPORT, and OE.SCP.IC apply for these objectives.
- T.Physical was refined to cover additional aspects of O.SCP.IC not contained in [1]. Therefore, the justification from [1] was adapted.

In the following the justifications are given.

O.PROTECT_DATA addresses the protection of the sensitive information (User Data or TSF data) stored in memories against unauthorized access. The TOE shall ensure that sensitive information stored in memories is protected against unauthorized disclosure and any corruption or unauthorized modification, which covers ***T.ACCESS_DATA*** and modification by unauthorized commands sequence, which covers partially ***T.OS_OPERATE***.

O.SIDE_CHANNEL addresses the protection of the security critical parts of the TOE and protects them from any disclosure by interpretation of physical or logical behavior based on leakage observation (e. g. side channel attacks). Especially, the NOS must be designed to avoid interpretations of electrical signals from the hardware part of the TOE. These characteristics cover the currents, voltages, power consumption, radiation, or timing of signals during the processing activity of the TOE. It allows covering entirely ***T.LEAKAGE***.

O.OPERATE and **O.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.

O.OS_DECEIVE addresses directly the threat ***T.OS_DECEIVE*** by ensuring that any loss of integrity cannot endanger the security, especially in case of modification of system flags or security attributes (e.g. cryptographic keys). The TOE shall prevent the fraudulent modification of such information as indicators or flags in order to go backwards, through the card life cycle sequence to gain access to prohibited information. The TOE must also guarantee the integrity of the NOS to prevent the modification of its expected behavior (for instance, code patch or rewriting).

O.FAULT_PROTECT ensures the correct continuation of operation of its security functions, in case of interruptions or changes carried out by physical actions (statically or dynamically). The TOE must ensure its correct operation even outside the normal operating conditions where reliability and secure operation has not been proven or tested. This is to prevent errors. The environmental conditions may include voltage, clock frequency, temperature, or external energy fields that can be applied on all interfaces of the TOE (physical or electrical). This addresses directly ***T.FAULT***.

O.PHYSICAL and **O.SCP.IC** ensures protection against physical manipulation of the IC, including the NOS and its application data (TSF data and User data). It prevents from disclose/modify TOE security features or functions provided by the IC. This covers ***T.PHYSICAL***.

O.IDENTIFICATION ensures that the TOE can be uniquely identified. This covers **OSP.PROCESS-TOE**.

OE.USE_DIAG ensures that secure communication protocols and procedures are used between the Smart Card and the terminal during phase 7 and helps to materialize ***A.USE_DIAG***.

OE.USE_KEYS ensures that the keys that stored by terminals or system outside the TOE control are protected in confidentiality. This objective is directly traced back to ***A.USE_KEYS***.

The objective **OE.NO-DELETION** ensures that the environmental assumption **A.NO-DELETION** is upheld. The objective **OE.NO-INSTALL** upholds the assumption **A.NO-INSTALL.**

4.3.1 Justifications from [1]

Confidentiality & Integrity

These are generic threats on code and data of *Java Card System* and *applet*s: *T.CONFID-JCS-CODE*, *T.CONFID-APPLI-DATA*, *T.CONFID-JCS-DATA*, *T.INTEG-APPLI-CODE*, *T.INTEG-JCS-CODE*, *T.INTEG-APPLI-DATA*, and *T.INTEG-JCS-DATA*.

Threats concerning the integrity and confidentiality of code are countered by the list of properties described in the (#.*VERIFICATION*) security issue. 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 visibility. As none of those instructions enables to read or modify a piece of code, no Java Card applet can therefore be executed to disclose or modify a piece of code. Native applications are also harmless because of the assumption (A.NATIVE), so no application can be run to disclose or modify a piece of code.

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

The threats concerning confidentiality and integrity of data are countered by bytecode verification and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of *applets* stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

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

Concerning the confidentiality and integrity of application sensitive data, as *applet*s may need to share some data or communicate with the *CAD*, cryptographic functions are required to actually protect the exchanged information (*O.*CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the *applet*s to use them. Keys and PIN's are particular cases of an application's sensitive data[6] that ask for appropriate management (*O.*KEY-MNGT, *O.*PIN-MNGT, *O.*TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) is also concerned.

Other application data that is sent to the *applet* as clear text arrives to the *APDU buffer*, which is a resource shared by all applications. The disclosure of such kind of data is prevented by the (*O.SHRD_VAR_CONFID*) security objective. The integrity of the information stored in that buffer is ensured by the (*O.SHRD_VAR_INTEG*) objective.

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

Identity Usurpation

T.SID.1	As impersonation is usually the result of successfully disclosing and modifying some assets, this threat is mainly countered by the objectives concerning the isolation of application data (like PINs), ensured by the (O.FIREWALL).

[6] The *Java Card System* may possess keys as well.

	Uniqueness of subject-identity (O.SID) also participates to face this threat. Note 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 OE.NO-INSTALL: applets are always installed in a secured environment that prevents any malevolent manipulation of the applets and cards.
T.SID.2	This is covered by integrity of TSF data, subject–identification (O.SID), the *firewall* (O.FIREWALL) and its good working order (O.OPERATE).

Unauthorized Executions

T.EXE-CODE.1	Unauthorized execution of a method is prevented by the objective OE.VERIFICATION. This threat particularly concerns the point (8) of the security issue (access modifiers and scope of visibility for classes, fields and methods). The O.FIREWALL objective is also concerned, because it prevents the execution of non-shareable methods of a class instance by any subject apart from the class instance owner.
T.EXE-CODE.2	Unauthorized execution of a method fragment or arbitrary data is prevented by the objective OE.VERIFICATION. This threat particularly concerns those points of the security issue related to control flow confinement and the validity of the method references used in the bytecodes.

Denial of Service

T.RESOURCES	An attacker prevents correct operation of the *Java Card System* through consumption of some resources of the card. This is directly countered by objectives on resource-management (O.RESOURCES) for runtime purposes and good working order (O.OPERATE) in a general manner. Note that, for what relates to CPU usage, the Java Card platform is single–threaded and it is possible for an ill–formed application (either native or not) to monopolize the CPU. However, a smart card can be physically interrupted (card removal or hardware reset) and most CADs implement a timeout policy that prevents them from being blocked should a card fail to answer.

The objective *O.CARD-MANAGEMENT* supports *OE.VERIFICATION* and contributes to cover all the threats on confidentiality and integrity of code and data. The objective also contributes, by preventing usurpation of identity resulting from a malicious installation of an applet on the card, to counter the threat *T.*SID*.1*.

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

The objective *OE.NATIVE* ensures that the environmental assumption A.NATIVE is upheld. The objective *OE.VERIFICATION* upholds the assumption A.VERIFICATION.

The following security objectives of the TOE are related to the assumptions made for this configuration as follows:

O.FIREWALL	The controlled sharing of data owned by different applications assumes that the code of the applications is well typed (A.VERIFICATION). Secured installation ensures the correct initialization of TSF data such as the identity of the applications (A.NO-INSTALL).
O.SID	The correct identification of the applications depends on the assumptions stating that pre-issuance applications have been correctly installed (A.NO-INSTALL), and that those are exactly the applications that will be on the card (A.NO-DELETION).

4.3.2 Justification from [4]

O.RND covers ***T.RND*** because the objective is are stated in a way, which directly corresponds to the description of the threat. It is clear from the description of the objective, that the corresponding threat is removed if the objective is valid. More specifically, in every case the ability to use the attack method successfully is countered, if the objective holds.

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 [27].

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 Generation of random numbers	1

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* [3].

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 [13] 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 the Family FMT_LIM

This section has been taken over from the certified (BSI-PP-0035) Smartcard IC Platform Protection profile [27].

The family FMT_LIM describes the functional requirements for the Test Features of the TOE. The new functional requirements were defined in the class FMT because this class addresses the management of functions of the TSF. The examples of the technical mechanism used in the TOE show that no other class is appropriate to address the specific issues of preventing the abuse of functions by limiting the capabilities of the functions and by limiting their availability.

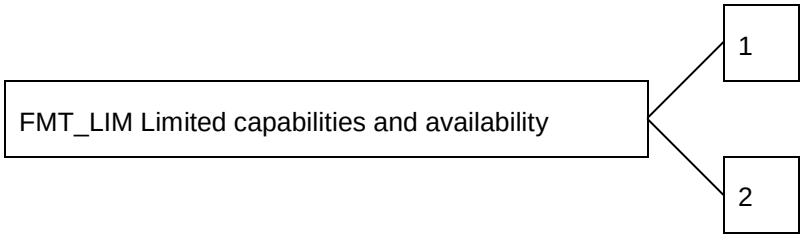
The family “Limited capabilities and availability (FMT_LIM)” is specified as follows.

FMT_LIM Limited capabilities and availability

Family behaviour

This family defines requirements that limit the capabilities and availability of functions in a combined manner. Note that FDP_ACF restricts the access to functions whereas the Limited capability of this family requires the functions themselves to be designed in a specific manner.

Component leveling:



FMT_LIM.1	Limited capabilities requires that the TSF is built to provide only the capabilities (perform action, gather information) necessary for its genuine purpose.
FMT_LIM.2	Limited availability requires that the TSF restrict the use of functions (refer to Limited capabilities (FMT_LIM.1)). This can be achieved, for instance, by removing or by disabling functions in a specific phase of the TOE’s life-cycle.
Management:	FMT_LIM.1, FMT_LIM.2 There are no management activities foreseen.
Audit:	FMT_LIM.1, FMT_LIM.2 There are no actions defined to be auditable.

To define the IT security functional requirements of the TOE an additional family (FMT_LIM) of the Class FMT (Security Management) is defined here. This family describes the functional requirements for the Test Features of the TOE. The new functional requirements were defined in the class FMT because this class addresses the management of functions of the TSF. The examples of the technical mechanism used in the TOE show that no other class is appropriate to address the specific issues of preventing the abuse of functions by limiting the capabilities of the functions and by limiting their availability.

The TOE Functional Requirement “Limited capabilities (FMT_LIM.1)” is specified as follows.

FMT_LIM.1	Limited capabilities
Hierarchical to:	No other components.
FMT_LIM.1.1	The TSF shall be designed in a manner that limits their capabilities so that in conjunction with “Limited availability (FMT_LIM.2)” the following policy is enforced [assignment: Limited capability and availability policy].
Dependencies:	FMT_LIM.2 Limited availability.

The TOE Functional Requirement “Limited availability (FMT_LIM.2)” is specified as follows.

FMT_LIM.2	Limited availability.
Hierarchical to:	No other components.
FMT_LIM.2.1	The TSF shall be designed in a manner that limits their availability so that in conjunction with “Limited capabilities (FMT_LIM.1)” the following policy is enforced [assignment: Limited capability and availability policy].
Dependencies:	FMT_LIM.1 Limited capabilities.
Application note:	The functional requirements FMT_LIM.1 and FMT_LIM.2 assume that there are two types of mechanisms (limited capabilities and limited availability) which together shall provide protection in order to enforce the policy. This also allows that • the TSF is provided without restrictions in the product in its user environment but its capabilities are so limited that the policy is enforced or conversely • the TSF is designed with high functionality but is removed or disabled in the product in its user environment. The combination of both requirements shall enforce the policy.

5.4 Definition of Family FAU_SAS

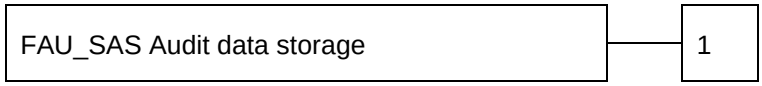
This section has been taken over from the certified (BSI-PP-0035) Smartcard IC Platform Protection profile [27].

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 defines the functional requirements for the TOE and the assurance requirements for the TOE. To define the functional requirements, users and subjects of the TOE are identified in the first section.

6.1 Users & Subjects

The life-cycle contains the following actors: (see also section 1.3.2)

- **NOS Developer:** includes the roles NOS developer and application developer (phase 1).
- **IC Manufacturer** includes the roles IC designer (phase 2) and IC manufacturer (phase 3).
- **Card Manufacturer** includes the roles IC Packaging Manufacturer and Smartcard Product Manufacturer and is responsible for IC Packaging and finishing process (phases 4-5)

The following description is taken over from Protection profile [1]:

*Subject*s 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) and software components (like the application *package*s 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 or in the case of this ST, the IC manufacturer.

The main *subject*s of the TOE considered in this document are the following ones:

- *Package*s used on the Java Card platform that act on behalf of the applet developer. These subjects are involved in the ***FIREWALL security policy*** defined in §6.2.1 and they should be understood as instances of the subject `S.PACKAGE`.
- The *CardManager,* can be considered a special instance of S.PACKAGE which implements the Open Platform specification. This package provides the functionality of a runtime environment running at the JCRE 'system' (privileged) context and for clarity is always represented by the subject `S.PACKAGE(CM)`.
- The *JCRE*, which acts on behalf of the card issuer. This subject is involved in several of the security policies defined in this document and is always represented by the subject `S.JCRE`.

Note: The subjects from [1]:

- (on-card) bytecode verifier
- installer
- applet deletion manager.

are not relevant for the minimal configuration and for this ST.

6.2 Security functional requirements (SFRs)

This section defines the functional requirements for the TOE using only functional requirements components drawn from the CC part 2 except four additional SFR (FCS_RNG.1, FMT_LIM.1, FMT_LIM.2, FPT_EMSEC.1) not contained in CC part 2. The functional requirements taken from [1] can be found in sections 6.2.1 - 6.2.6. Further Functional Requirements not contained in [1] can be found in section 6.2.7. The Functional Requirements in 6.2.7.11, 6.2.7.12, and 6.2.7.13 are newly defined.

The permitted operations (assignment, iteration, selection and refinement) of the SFR related to Common Criteria [13] and extended SFRs defined in chapter 5 are printed in ***bold and italics*** type. Completed operations related to the PP are additionally marked within **[ ]**. Editorial changes are printed in *italics*. If appropriate, an explanatory note on the operation is given, e. g. to justify a difference to the SFR as defined in the underlying PP.

The prefix used to introduce objects is "OB". This was done to avoid confusions with the TOE objectives.

The following table gives an overview of the used SFR for the TOE. Dependencies being not fulfilled are printed in *italics*. These are either explained below the table or alternatives.

Table 18. TOE security functionality requirements

Class / Component	Name	Hierar.	Dependency

Class / Component	Name	Hierar.	Dependency
FAU	**Security audit**		
FAU_ARP.1/ JCS	Security alarms	no	FAU_SAA.1
FAU_SAA.1	Potential violation analysis	no	*FAU_GEN.1, see below*
FAU_SAS.1/ SCP	Audit data storage	no	no
FCS	**Cryptographic support**		
FCS_CKM.1	Cryptographic key generation	no	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4
FCS_CKM.2	Cryptographic key distribution	no	[FDP_ITC.1 *or FDP_ITC.2* or FCS_CKM.1] FCS_CKM.4
FCS_CKM.3	Cryptographic key access	no	[FDP_ITC.1 *or FDP_ITC.2* or FCS_CKM.1] FCS_CKM.4
FCS_CKM.4	Cryptographic key destruction	no	[FDP_ITC.1 *or FDP_ITC.2* or FCS_CKM.1]
FCS_COP.1	Cryptographic operation	no	[FDP_ITC.1 *or FDP_ITC.2* or FCS_CKM.1] FCS_CKM.4
FCS_RNG.1	Quality metric for random numbers	no	no
FDP	**User data protection**		
FDP_ACC.1/ CMGR	Subset access control	no	FDP_ACF.1/CMGR
FDP_ACC.1/ SCP	Subset access control	no	FDP_ACF.1/SCP
FDP_ACC.2/ FIREWALL	Complete access control	FDP_ACC.1	FDP_ACF.1/FIREWALL
FDP_ACF.1/ CMGR	Security attribute based access control	no	FDP_ACC.1/CMGR, FMT_MSA.3/CMGR
FDP_ACF.1/ FIREWALL	Security attribute based access control	no	FDP_ACC.1/ CMGR, FMT_MSA.3/ CMGR
FDP_ACF.1/ SCP	Security attribute based access control	no	FDP_ACC.1/SCP, FMT_MSA.3/SCP
FDP_ETC.1	Export of user data without security attributes	no	[*FDP_ACC.1 or* FDP_IFC.1/JCVM]

Class / Component	Name	Hierar.	Dependency
FDP_IFC.1/ JCVM	Subset Information flow control	no	FDP_IFF.1/JCVM
FDP_IFC.1/ SCP	Subset information flow control	no	*FDP_IFF.1, see below*
FDP_IFF.1/ JCVM	Simple security attributes	no	FDP_IFC.1/JCVM, FMT_MSA.3/FIREWALL
FDP_ITC.1	Import of user data without security attributes	no	[*FDP_ACC.1 or* FDP_IFC.1/JCVM] FMT_MSA.3/CMGR
FDP_ITT.1/ SCP	Basic internal transfer protection	no	[*FDP_ACC.1 or* FDP_IFC.1/SCP]
FDP_RIP.1	Subset residual information protection	no	no
FDP_ROL.1/ FIREWALL	Basic rollback	no	[FDP_ACC.2/FIREWALL *or FDP_IFC.1]*
FDP_SDI.2	Stored data integrity monitoring and action	FDP_SDI.1	no
FIA	**Identification and authentication**		
FIA_AFL.1	Authentication failure handling	no	FIA_UAU.1
FIA_ATD.1/ AID	User attribute definition	no	no
FIA_UAU.1	Timing of authentication	no	FIA_UID.1/CMGR
FIA_UAU.3/ CMGR	Unforgeable authentication	no	no
FIA_UAU.4/ CMGR	Single-use authentication mechanisms	no	no
FIA_UID.1/ CMGR	Timing of identification	no	no
FIA_UID.2/AID	User identification before any action	FIA_UID.1	no
FIA_USB.1	User-subject binding	no	FIA_ATD.1
FMT	**Security management**		
FMT_LIM.1	Limited capabilities	no	FMT_LIM.2
FMT_LIM.2	Limited availability	no	FMT_LIM.1
FMT_MSA.1/ CMGR	Management of security attributes	no	[FDP_ACC.1/CMGR *or FDP_IFC.1*] FMT_SMF.1,

Class / Component	Name	Hierar.	Dependency
			FMT_SMR.1/CMGR
FMT_MSA.1/ JCRE	Management of security attributes	no	[FDP_ACC.1/ FIREWALL *or FDP_IFC.1*] FMT_SMF.1, FMT_SMR.1/CMGR
FMT_MSA.2/ JCRE	Secure security attributes	no	[FDP_ACC.1/CMGR *or FDP_IFC.1*] FMT_MSA.1/JCRE, FMT_SMR.1/CMGR
FMT_MSA.3/ CMGR	Static attribute initialization	no	FMT_MSA.1/CMGR, FMT_SMR.1/CMGR
FMT_MSA.3/ FIREWALL	Static attribute initialization	no	FMT_MSA.1/JCRE, FMT_SMR.1/JCRE
FMT_MSA.3/ SCP	Static attribute initialization	no	FMT_MSA.1/SCP, FMT_SMR.1/JCRE
FMT_MTD.1/ JCRE	Management of TSF data	no	FMT_SMF.1, FMT_SMR.1/JCRE
FMT_MTD.3	Secure TSF data	no	FMT_MTD.1/JCRE
FMT_SMF.1	Specification of Management Functions	no	no
FMT_SMR.1/ JCRE	Security roles	no	FIA_UID.2/AID
FMT_SMR.1/ CMGR	Security roles	no	FIA_UID.1/CMGR
FPR	**Privacy**		
FPR_UNO.1	Unobservability	no	no
FPT	**Protection of the TSF**		
FPT_EMSEC	TOE Emanation	no	no
FPT_FLS.1/ JCS	Failure with preservation of secure state	no	no
FPT_FLS.1/ SCP	Failure with preservation of secure state	no	no
FPT_ITT.1/ SCP	Basic internal TSF data transfer protection	no	no
FPT_PHP.1	Passive detection of physical attack	no	no
FPT_PHP.3/ SCP	Resistance to physical attack	no	no

Class / Component	Name	Hierar.	Dependency
FPT_RCV.3/ SCP	Trusted Recovery	FPT_RCV.2	AGD_OPE.1
FPT_RCV.4/ SCP	Trusted Recovery	no	no
FPT_TDC.1	Inter-TSF basic TSF data consistency	no	no
FPT_TST.1	TSF testing	no	no
FRU	**Resource utilization**		
FRU_FLT.2 /SCP	Limited fault tolerance	FRU_FLT.1	FPT_FLS.1/SCP
FTP	**Trusted path/channels**		
FTP_ITC.1/ CMGR	Inter-TSF trusted channel	no	no

The following dependencies are not fulfilled:

1. From FAU_SAA.1 to FAU_GEN.1

The dependency of FAU_SAA.1 with FAU_GEN.1 is not applicable to the TOE; the FAU_GEN.1 component forces many security relevant events to be recorded (due to dependencies with other functional security components) and this is not achievable in a Smart Card since many of these events result in card being in an insecure state where recording of the event itself could cause a security breach. It is then assumed that the function FAU_SAA.1 may still be used and the specific audited events will have to be defined in the ST independently with FAU_GEN.1.

2. From FDP_IFC.1/SCP to FDP_IFF.1

Thespecification of FDP_IFF.1 would not capture the nature of the security functionalrequirement nor add any detail. As stated in the Data Processing Policy referred to in FDP_IFC.1 there are no attributes necessary. The security functional requirement for the TOE is sufficiently described using FDP_ITT.1 and its Data processing Policy (FDP_IFC.1). Therefore the dependency is considered satisfied. (see [4]).

Additionally, the extended SFR FAU_SAS.1, FCS_RNG.1, FMT_LIM.1 and FMT_LIM.2 have been taken over from [27] and FPT_EMSEC.1 has been taken over from the certified (BSI-PP-0017) Protection Profile Machine Readable travel Document with “ICAO Application”, Basic Access Control [3].

6.2.1 Firewall Policy

6.2.1.1 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*, *OB.JAVAOBJECT* and all operations among subjects and objects covered by the SFP.

Subjects (prefixed with an "*S*") and objects (prefixed with an "*OB*") covered by this policy are:

Table 19. Subjects and Objects for FDP_ACC.2.1/FIREWALL

Table description (optional)

Subject / Object	Description
S.PACKAGE	Any *package*, which is the security unit of the firewall policy.
S.JCRE	The *JCRE*. This is the process that manages *applet* selection and de-selection, along with the delivery of *APDU*s from and to the smart card device. This subject is unique.
OB.JAVAOBJECT	Any object. Note that KEYS, PIN, arrays and *applet* instances are specific objects in the Java programming language.

Operations (prefixed with "*OP*") of this policy are described in the following table. Each operation has a specific number of 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 20. Operations for FDP_ACC.2.1/FIREWALL

Operation	Description
OP.ARRAY_ACCESS(OB.JAVAOBJECT, field*)*	Read/Write an array component.
OP.INSTANCE_FIELD(OB.JAVAOBJECT, field*)*	Read/Write a field of an instance of a class in the Java programming language
OP.INVK_VIRTUAL(OB.JAVAOBJECT, method, arg_1,…*)*	Invoke a virtual method (either on a class instance or an array object)
OP.INVK_INTERFACE(OB.JAVAOBJECT, method, arg_1,…*)*	Invoke an *interface* method.
OP.THROW(OB.JAVAOBJECT)	Throwing of an object (**athrow**).
OP.TYPE_ACCESS(OB.JAVAOBJECT, class*)*	Invoke **checkcast** or **instanceof** on an object.
OP.JAVA(…)	Any access in the sense of [8], §6.2.8. In our formalization, this is one of the preceding operations.
*OP.CREATE(*Sharing, LifeTime*)*	Creation of an object (**`new`** or **`makeTransient`** call).

Note that accessing array's components of a **static** array, and more generally fields and methods of **static** objects, is an access to the corresponding *OB.JAVAOBJECT*.

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.

Editorial changes: The phrase "subject in the TSC and any object within the TSC" in [1] was replaced by "subject controlled by the TSF and any object controlled by the TSF" according to Common Criteria 3.1 [13].

6.2.1.2 FDP_ACF.1/FIREWALL Security Attribute based Access Control

See FMT_MSA.1 for more information about security attributes.

FDP_ACF.1.1/FIREWALL The TSF shall enforce the ***FIREWALL* access control SFP** to objects based on *the following*: **the security attributes of the covered subjects and objects contained in the following tables.**

Editorial changes:

1.) The word "the following" results from final interpretation FI103 which was not considered in [4].

2.) The assignment contained in [1] "(1) the security attributes of the covered subjects and objects, (2) the currently active context and (3) the SELECTed applet context" has been changed according to F103, because attributes must be assigned to either subjects or objects.

The following table describes which security attributes are attached to which subject/object of our policy.

Table 21. Attributes of Subjects and objects for FDP_ACF.1.1/FIREWALL

Subject/Object	Attributes
S.PACKAGE	Context
S.JCRE	None
OB.JAVAOBJECT	Sharing, Context, LifeTime, SELECTed applet Context[7]

The following table describes the possible values for each security attribute.

Table 22. possible values for each security attribute (FDP_ACF.1.1/FIREWALL)

Name	Description
Context	*Package AID*, or "JCRE"
Sharing	Standard, SIO, JCRE entry point, or global array
LifeTime	**CLEAR_ON_DESELECT** or **PERSISTENT**.[8]

[7] The security attribute SELECTed applet Context was assigned to object OB.JAVAOBJECT according to final interpretation FI103.

Name	Description
SELECTed applet Context	*Package AID*, or “None”

In the case of an array type, we state that fields are components of the array ([12], §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 ([8], §6.1.2). 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).

- ([8], Glossary) *Currently selected applet***.** The JCRE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command 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.

While the expression “selected applet” refers to a specific installed applet, the relevant aspect to the policy is the *context* of the selected *applet*; that is why the associated security attribute is a *package AID*.

- ([8], §6.1.1) At any point in time, there is only **one active context** within the VM (this is called the currently active context).

This should be identified in our model with the acting `S.PACKAGE`’s context (see “*Currently context*” in the glossary). This value is in one-to-one correspondence with *AIDs* of *package*s (except for the JCRE context, of course), which appears in the model in the “Context” attribute of both subjects and objects of the policy. The reader should note that the invocation of **static** methods (or access to a **static** field) is

[8] *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.

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 in this case.

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: *by the*[9] ***FIREWALL SFP****:*

- **R.JAVA.1** ([8] §6.2.8) An *S.PACKAGE* may freely perform any of *OP.ARRAY_ACCESS*,*OP.INSTANCE_FIELD*, *OP.INVK_VIRTUAL*, *OP.INVK_INTERFACE*, *OP.THROW* or *OP.TYPE_ACCESS* upon any *OB.JAVAOBJECT* whose Sharing attribute has value ***"JCRE entry point"*** or **"global array".**
- **R.JAVA.2** ([8] §6.2.8) An *S.PACKAGE* may freely perform any of *OP.ARRAY_ACCESS*,*OP.INSTANCE_FIELD*, *OP.INVK_VIRTUAL*, *OP.INVK_INTERFACE* or *OP.THROW* upon any *OB.JAVAOBJECT* whose Sharing attribute has value **"Standard"** and whose Lifetime attribute has value **"PERSISTENT"** only if *OB.JAVAOBJECT*'s Context attribute has the same value as the active context.
- **R.JAVA.3** ([8] §6.2.8.10) An *S.PACKAGE* may perform *OP.TYPE_ACCESS* upon an *OB.JAVAOBJECT* whose Sharing attribute has value **"SIO"** only if *OB.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** ([8] §6.2.8.6) An *S.PACKAGE* may perform *OP.INVK_INTERFACE* upon an *OB.JAVAOBJECT* whose Sharing attribute has the value **"SIO"** only if the invoked *interface* method extends the **Shareable** *interface*.
- **R.JAVA.5** An *S.PACKAGE* may perform an *OP.CREATE* only if the value of the Sharing parameter[10] is "Standard".

At last, rules governing access to and creation of OB.JAVAOBJECTs by S.JCRE are essentially implementation-dependent (however, see FDP_ACF.1.3/FIREWALL.)

FDP_ACF.1.3/FIREWALL The TSF shall explicitly authorize access of subjects to objects based on the following additional rule[11]:

9 Editorial Change: The word "by the" was introduced as as proposed in the PP.

10 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.

11 Editorial Change: The word "rules" in CC part 2 was replaced by "rule" as proposed in the PP, because only one rule is added.

The subject **S.JCRE** can freely perform **OP.JAVA(…)** and **OP.CREATE**, with the exception given in ***FDP_ACF.1.4/FIREWALL***.

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

- Access shall be denied by any subject with *OP.JAVA* upon an *OB.JAVAOBJECT* whose LifeTime attribute has value "**CLEAR_ON_DESELECT**" if ***OB.JAVAOBJECT***'s Context attribute is not the same as the SELECTed applet Context.
- Access shall be denied by any subject with *OP.CREATE* and a "**CLEAR_ON_DESELECT**" LifeTime parameter if the active context is not the same as the SELECTed applet Context.

Application Note: The deletion of applets may render some OB.JAVAOBJECT inaccessible, and the JCRE 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. Such a mechanism is implementation-dependent.

Note: For this Java Card no applet deletion is possible, therefore this application note is not relevant for this evaluation.

6.2.1.3 FDP_IFC.1/JCVM Subset Information Flow Control

FDP_IFC.1.1/JCVM — The TSF shall enforce the ***JCVM information flow control SFP*** on ***the following subjects, information and operations***.

Subjects[12] (prefixed with an "*S*") and information (prefixed with an "*I*") covered by this policy are:

Table 23. Subject and Information for FDP_IFC.1.1/JCVM

Subject/Information	Description
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	Any object's field, **static** field or array position.
I.DATA	JCVM Reference Data: objectref addresses of temporary JCRE Entry Point objects and global arrays.

There is a unique operation in this policy:

Table 24. Operations for FDP_IFC.1.1/JCVM

Operation	Description
OP.PUT(S1, S2, I)	Transfer a piece of information I from S1 to S2.

Application Note: References of temporary JCRE entry points, which cannot be stored in class variables, instance variables or array

[12] Information flow policies control the flow of information between "subjects". This is a purely terminological choice; those "subjects" can merely be passive containers. They are not to be confused with the "active entities" of access control policies.

components, are transferred from the internal memory of the JCRE (TSF data) to some stack through specific APIs (JCRE owned exceptions) or JCRE invoked methods (such as the ***process(APDU apdu)****); these are causes of* `OP.PUT(S1,S2,I)` *operations as well.*

6.2.1.4 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: subjects: S.LOCAL, S.MEMBER; information I.DATA; attribute: the currently active context.
Refinement:	Due to application of final interpretation FI104 the assignment must contain the list of subjects, information, and corresponding security attributes.
FDP_IFF.1.2/JCVM	The TSF shall permit an information flow between a controlled subject and controlled information *through*[13] a controlled operation if the following *rule holds*[14]: **An operation *OP.PUT(S1, S.MEMBER, I)* is allowed if and only if the active context is "JCRE"; other *OP.PUT* operations are allowed regardless of the active context's value.**
FDP_IFF.1.3/JCVM	The TSF shall enforce **[*no additional information flow control SFP rules*]**.
FDP_IFF.1.4/JCVM	The TSF shall explicitly authorize an information flow based on the following rules: **[*none*]**.
FDP_IFF.1.5/JCVM	The TSF shall explicitly deny an information flow based on the following rules: **[*none*]** *Application Note: the Storage of temporary JCRE-owned objects' references is runtime-enforced ([8], §6.2.8.1-3).* *Editorial changes: FDP_IFF.1.4/JCVM ("The TSF shall provide the following [no additional SFP capabilities].)" in [1] was removed according to Common Criteria 3.1 [13].*

Note that this policy essentially applies to the execution of bytecode. Native methods[15], the JCRE itself and possibly some API methods can be granted specific rights or limitations through the ***FDP_IFF.1.3/JCVM*** to ***FDP_IFF.1.5/JCVM*** elements. The way the virtual machine manages the transfer of values on the stack and local variables (returned values, uncaught exceptions) from and to internal registers is implementation-dependent. For instance, a returned reference, depending on the implementation of the stack frame, may transit trough an internal register prior to being pushed on the stack of the invoker. The **areturn** bytecode would cause more than one *OP.PUT* operation under this scheme.

[13] Editorial change: The word "via" was replaced by "through" as proposed in the PP.

[14] Editorial change: The word "rules hold" in CC part 2 was replaced by "rule holds", as only one rule is added.

[15] Note: For this TOE, they are no native methods.

6.2.1.5 FDP_RIP.1 Subset Residual Information Protection

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

Application Note: The semantics of the Java programming language requires for any object field and array position to be initialized with default values when the resource is allocated [12],§2.5.1.

6.2.1.6 FMT_MSA.1/JCRE Management of Security Attributes

(See FMT_SMR.1.1/JCRE for the roles)

FMT_MSA.1.1/JCRE The TSF shall enforce ***the FIREWALL access control SFP and the JCVM information flow control SFP*** to restrict the ability to ***modify*** *the* ***active context and the SELECTed applet Context*** *security attributes*[16] *to* ***the JCRE*** **(***S.JCRE***).**

Application Note: The modification of the active context as well as that of the selected applet should be performed in accordance with the rules given in [8], §4 and [9], §3.4.

6.2.1.7 FMT_MSA.2/JCRE Secure Security Attributes

FMT_MSA.2.1/JCRE The TSF shall ensure that only secure values are accepted for

- **the Context attribute of a **.JAVAOBJECT*[17] 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 and**
- **any *OB.JAVAOBJECT* whose LifeTime attribute value is not** PERSISTENT **has an array type as JavaCardClass attribute's value**.

Application Note: The last two rules are motivated by the fact that the Java Card API defines only transient arrays factory methods. Future versions may allow the creation of transient objects belonging to arbitrary classes; such evolution will naturally change the range of "secure values" for this component.

Editorial changes: The phrase "security attributes" in [1] was replaced by the list of security attributes in the Application note according to Common Criteria 3.1 [13].

16 Editorial change for better readability of the sentence.

17 Either subject or object.

6.2.1.8 FMT_MSA.3/FIREWALL Static Attribute Initialization

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

Application Note: 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 operation is permitted by the SFP, 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 ([8], §6.1.2). There is one default value for the SELECTed applet Context that is the default applet identifier's Context, and one default value for the active context, that is "JCRE".

Application Note: There is no security attribute attached to subjects or information for this information flow policy. However, this is the JCRE who controls the currently active context. Moreover, 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 JCRE (and the virtual machine).

FMT_MSA.3.2/FIREWALL The TSF shall allow the ***following role(s)*** to specify alternative initial values to override the default values when an object or information is created: ***none***[18]***.***

*Application Note: The intent is that none of the identified roles has privileges with regard to the default values of the security attributes. Notice that creation of objects is an operation controlled by the **FIREWALL SFP**; the latitude on the parameters of this operation is described there. The operation shall fail anyway if the created object would have had security attributes whose value violates **FMT_MSA.2.1/JCRE**.*

6.2.1.9 FMT_SMR.1/JCRE Security roles

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

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

6.2.1.10 TSF domain separation

Editorial changes: FPT_SEP.1.1 and FPT_SEP.1.2 in [1] were removed according to Common Criteria 3.1 [13].

6.2.2 Application Programming Interface

The following SFRs are related to the Java Card API and extensions like Proprietary BAC Accelerator Interface.

[18] Editorial Change: The sentence was changed for better readability.

6.2.2.1 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 **[*JCOP K3-RNG*]** and specified cryptographic KEY sizes **[*DES: 112, 168 Bit, RSA: 1976 - 2048 Bit, EC: 224-320, AES: 128, 192, 256 Bit*]** that meet the following: **[[32], *[31] section 7.2 for EC key generation*]**.

Application Note: The keys can be generated and diversified in accordance with [8] specification in classes ***KeyBuilder*** *and* ***KeyPair*** *(at least Session key generation).*

6.2.2.2 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 **[*methods: setKey for DES and AES, setExponent and setModulus for RSA, as well as, setA, setB, setFieldFP, setG, setK, and setR for EC*]** that meets the following: **[*[8]*]**.

6.2.2.3 FCS_CKM.3 Cryptographic KEY Access

FCS_CKM.3.1 The TSF shall perform **[*management of DES, AES, EC, and RSA-keys*]** in accordance with a specified cryptographic key access method **[*methods/commands defined in packages javacard.security and javacardx.crypto of [8]*]** that meets the following: **[*[8]*]**.

Application Note: The keys can be accessed in accordance with [8] in class ***Key****.*

6.2.2.4 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 **[*physically overwriting the keys* with zeros *by method* (e.g. *clearKey of* [8]) *or automatically on applet deselection*]** that meets the following: **[none]**.

Application Note: The keys are reset in accordance with [8] in class ***Key*** *with the method* ***clearKey()****. Any access to a cleared key attempting to use it for ciphering or signing shall throw an exception.*

6.2.2.5 FCS_COP.1 Cryptographic Operation

FCS_COP.1/TripleDES

FCS_COP.1.1 The TSF shall perform **[*data encryption and decryption*]** in accordance with a specified cryptographic algorithm **[*Triple-DES in ECB/CBC Mode*]** and cryptographic key size **[*112, 168 Bit*]** that meet the following: **[[20], pg 8ff]**.

FCS_COP.1/AES

FCS_COP.1.1 The TSF shall perform **[*data encryption and decryption*]** in accordance with a specified cryptographic algorithm **[*AES in ECB/CBC Mode*]** and cryptographic key size **[*128, 192, and 256 Bit*]** that meet the following: [**[19], section 5**].

FCS_COP.1/RSACipher

FCS_COP.1.1	The TSF shall perform **[*data encryption and decryption*]** in accordance with a specified cryptographic algorithm **[*RSA*]** and cryptographic key size **[*1976 - 2048 Bit*]** that meet the following: [**[16], section 5.1.1 and 5.1.2**]**.**

FCS_COP.1/DHKeyExchange

FCS_COP.1.1	The TSF shall perform **[*Diffie-Hellman key agreement*]** in accordance with a specified cryptographic algorithm **[*ECC over GF(p)*]** and cryptographic key size **[*224 - 320 Bit*]** that meet the following: **[**[26] **section 6.3] .**

FCS_COP.1/DESMAC

FCS_COP.1.1	The TSF shall perform **[*8 byte MAC generation and verification*]** in accordance with a specified cryptographic algorithm **[*Triple-DES in outer CBC Mode*]** and cryptographic key size **[*112, 168 Bit*]** that meet the following: **[[24], section 6 and 7]**.

FCS_COP.1/RSASignatureISO9796

FCS_COP.1.1	The TSF shall perform **[*digital signature generation and verification*]** in accordance with a specified cryptographic algorithm **[*RSA with SHA-1]*** and cryptographic key size **[*1976 - 2048 Bit*]** that meet the following: **[[23], section 8]**.

FCS_COP.1/RSASignaturePKCS#1

FCS_COP.1.1	The TSF shall perform **[*digital signature generation and verification*]** in accordance with a specified cryptographic algorithm **[*RSA with SHA-1*]** and cryptographic key size **[*1976 - 2048 Bit*]** that meet the following: [**[16] , section 5.1.1 and 5.1.2**]**.**

FCS_COP.1/ECSignature

FCS_COP.1.1	The TSF shall perform **[*digital signature generation and verification*]** in accordance with a specified cryptographic algorithm **[*EC with SHA-1 SHA-224, and SHA-256*]** and cryptographic key size **[*224 - 320 Bit*]** that meet the following: **[**[25] **section 6.4].**

FCS_COP.1/ECAdd

FCS_COP.1.1	The TSF shall perform **[*secure point addition*]** in accordance with a specified cryptographic algorithm **[*ECC over GF(p)*]** and cryptographic key size **[*224 - 320 Bit*]** that meet the following: **[[25] Annex C].**

FCS_COP.1/SHA-1

FCS_COP.1.1	The TSF shall perform **[*secure hash computation*]** in accordance with a specified cryptographic algorithm **[*SHA-1*]** and cryptographic key size **[*none*]** that meet the following: [**[18]section 6**]**.**

FCS_COP.1/SHA-224

FCS_COP.1.1	The TSF shall perform **[*secure hash computation*]** in accordance with a specified cryptographic algorithm **[*SHA-224*]** and cryptographic key size **[*none*]** that meet the following: ***[[18] section 6]*.**

FCS_COP.1/SHA-256

FCS_COP.1.1	The TSF shall perform **[*secure hash computation*]** in accordance with a specified cryptographic algorithm **[*SHA-256*]** and cryptographic key size **[*none*]** that meet the following: **[*[18] section 6*].**

FCS_COP.1/TDES_MRTD

FCS_COP.1.1	The TSF shall perform **[*secure messaging – encryption and decryption*]** in accordance with a specified cryptographic algorithm **[*Triple-DES in CBC mode*]** and cryptographic key sizes **[*112 bit*]** that meet the following: **[*[20] pg 8 ff*]**.

FCS_COP.1/MAC_MRTD

FCS_COP.1.1	The TSF shall perform **[*secure messaging – message authentication code*]** in accordance with a specified cryptographic algorithm **[*Retail MAC*]** and cryptographic key sizes **[*112 bit*]** that meet the following: **[[[24], section 6 and 7]**.

6.2.2.6 FDP_RIP.1 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 *object*[19]: ***the APDU buffer.*** *Application Note: The allocation of a resource to the APDU buffer is typically performed as the result of a call to the* `process()` *method of an applet.*
FDP_RIP.1.1/bArray	The TSF shall ensure that any previous information content of a resource is made unavailable upon the ***de-allocation*[20] *of the resource from*** the following *object*[19]*:* ***the bArray object***. *Application Note: A resource is allocated to the bArray object when a call to an applet's* `install()` *method is performed. There is no conflict with FDP_ROL.1 here because of the bounds on the rollback mechanism (**FDP_ROL.1.2/FIREWALL**): the scope of the rollback does not extend outside the execution of the* `install()` *method, and the de-allocation occurs precisely right after the return of it.*
FDP_RIP.1.1/TRANSIENT	The TSF shall ensure that any previous information content of a resource is made unavailable upon the ***de-allocation***[20] ***of the resource from*** the following objects*:* ***any transient object***. *Application Note: The events that provoke the de-allocation of a transient object are described in [8], §5.1.*

[19] Editorial Change: The word "objects" was replaced by "object" as proposed in the PP.

[20] Editorial Change: The word "deallocation" was replaced by "de-allocation" as proposed in the PP.

FDP_RIP.1.1/ABORT The TSF shall ensure that any previous information content of a resource is made unavailable upon the ***de-allocation***[20] ***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 the previously mentioned references are described in [8], §7.6.3.

FDP_RIP.1.1/KEYS The TSF shall ensure that any previous information content of a resource is made unavailable upon the ***de-allocation***[20] ***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 [7].*

Application Note: Java Card System 2.1.1 defines no explicit (or implicit) de-allocation of objects, but those caused by the failure of installation or the abortion of a transaction. The only related function for keys is the `clearKey()` *method, which does not mandate erasure of the contents of the key (see* ***FCS_CKM.4****) nor the behavior of the transaction with respect to this "clearing". ST authors may consider additional security requirements on this topic.*

6.2.2.7 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 ***OP.JAVA, OP.CREATE*** on[21] ***OB.JAVAOBJECTs***.

FDP_ROL.1.2/FIREWALL The TSF shall permit operations to be rolled back within the scope of a `select()`, `deselect()`, `process()` **or** `install()` ***call, notwithstanding the restrictions given in [8], §7.7, within the bounds of the Commit Capacity ([8], §7.8), and those described in [7]***.

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

The loading and linking of *applet package*s (the installation or registration is covered by ***FDP_ROL.1.1/FIREWALL***) is subject to some kind of rollback mechanism, described in [8], §10.1.4, but is implementation-dependent.

6.2.3 Card Security Management

The following SFRs are related to the security requirements at the level of the whole card, in contrast to the previous ones, that are somewhat restricted to the TOE alone. For

[21] Editorial change: Rewording for better understanding as proposed in the PP.

instance, a potential security violation detected by the virtual machine may require a reaction that does not only concern the virtual machine, such as blocking the card (or request the appropriate security module with the power to block the card to perform the operation).

6.2.3.1 FAU_ARP.1/JCS Security Alarms

FAU_ARP.1.1/JCS — The TSF shall *throw*[22] ***an exception, lock the card session or reinitialize the Java Card System and its data*** **[no other actions]** upon detection of a potential security violation.

REFINEMENT — Potential security violation is refined to one of the following events:

- *applet* life cycle[23] inconsistency
- Card tearing (unexpected removal of the Card out of the CAD) and power failure
- Abortion of a transaction in an unexpected context (see (`abortTransaction()`,[7] and ([8], §7.6.2)
- Violation of the Firewall or JCVM SFPs
- Unavailability of resources
- Array overflow
- Other runtime errors related to *applet*'s failure (like uncaught exceptions)
- **Card Manager life cycle state (FUSED, PROTECTED, 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.**
- **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**
- **Illegal access to the previously defined D.JAVA_OBJECT objects audited through the firewall mechanism** [24]

Application Note: The thrown exceptions and their related events are described in [7], [8], and [9].

Application Note: 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.

---

22 Editorial change: Rewording for better understanding as proposed in the PP.

23 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.

24 The events in printed in bold-style are additional events regarding to [PP0002]

*Application Note: 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).*

Application Note: 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 lasts up to the next "RESET" event, without affecting other components of the card (such as the card manager).

Finally, because the installation of *applet*s is a sensitive process, security alerts in this case should also be carefully considered herein.

6.2.3.2 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 **[*integrity errors*]** on all objects, based on the following attributes: **[*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 **[*maintain a secure state and return an error message*]**.

Editorial changes: The phrase "within the TSC" in [1] was replaced by "in containers controlled by the TSF" according to Common Criteria 3.1 [13].

6.2.3.3 FPT_RVM.1 Reference Mediation

Editorial changes: FPT_RVM.1 in [1] was removed according to Common Criteria 3.1 [13].

6.2.3.4 FPT_FLS.1/JCS Failure with Preservation of Secure State

FPT_FLS.1.1/JCS	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/JCS.***

The JCRE Context is the Current Context when the VM begins running after a card reset ([8], §6.2.3). Behavior of the TOE on power loss and reset is described in [8], §3.5, and §7.1.

6.2.3.5 FPR_UNO.1 Unobservability

FPR_UNO.1.1	The TSF shall ensure that **[*subjects S.Package*]** are unable to observe the operation **[*all operations*]** on **[*secret keys and PIN codes*]** by **[*other subjects S.Package*]**.

Application Note: Although it is not required in [8] 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.

6.2.3.6 FPT_TST.1 TSF Testing

FPT_TST.1.1 The TSF shall run a suite of self-tests ***at the conditions during initial start-up at each power on*** to demonstrate the correct operation of ***the TSF***[25].

Application Note: TSF-testing is not mandatory in [8], but appears in most of security requirements documents for masked applications. Testing could also occur randomly.

FPT_TST.1.2 The TSF shall provide authorized users with the capability to verify the integrity of ***TSF data***[26].

FPT_TST.1.3 The TSF shall provide authorized users with the capability to verify the integrity of ***TSF***.

6.2.4 AID Management

6.2.4.1 FMT_MTD.1/JCRE Management of TSF Data

(See FMT_SMR.1.1/JCRE for the roles)

FMT_MTD.1.1/JCRE The TSF shall restrict the ability to ***modify*** the ***list of registered applets' AID*** to **[*the JCRE only*]**.

*Application Note: The installer and the JCRE manage some other TSF data such as the applet life cycle or CAP files, but this management is implementation specific. Objects in the Java programming language may also try to query AIDs of installed applets through the **lookupAID(..)** API method.*

*Application Note: The installer, applet deletion manager or even the card manager may be granted the right to modify the list of registered applets' AIDs in specific implementations (possibly needed for installation and deletion; see #.**DELETION** and #.**INSTALL**).*

6.2.4.2 FMT_MTD.3 Secure TSF data

FMT_MTD.3.*1* The TSF shall ensure that only secure values are accepted for **[*Card Manager life cycle state (FUSED, PROTECTED, OP_READY, INITIALIZED, SECURED, CARD_LOCKED, TERMINATED*].**

Editorial changes: The phrase "security attributes" in [1] was replaced by a list of security attributes in the Application note according to Common Criteria 3.1 [13].

6.2.4.3 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: ***the AID and version number of each package, the AID of each registered applet, and***

[25] Editorial change according to final interpretation FI056 where "the TSF" is part of a selection operation.

[26] Editorial change according to final interpretation FI056 where "the TSF data" is part of a selection operation.

whether a registered applet is currently selected for execution ([9], §6.5).

6.2.4.4 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) which 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.

Application Note: The role JCRE defined in FMT_SMR.1/JCRE is attached to an IT security function rather than to a "user" of the CC terminology. The JCRE does not "identify" itself with respect to the TOE, but it is a part of it.

Editorial changes: The phrase "identify itself" in [1] was replaced by "to be successfully identified" according to Common Criteria 3.1 [13].

6.2.4.5 FIA_USB.1 User-Subject binding

Note: According to final interpretation #137 the SFR FIA_USB.1 is rewritten as:

FIA_USB.1.1 — The TSF shall associate the following user security attributes with subjects acting on the behalf of that user: **[*active Context and SELECTed applet Context security attribute*].**

FIA_USB.1.2 — The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **[*rules defined in FDP_ACF.1.1/FIREWALL, FMT_MSA.2.1/JCRE, and FMT_MSA.3.1/FIREWALL and corresponding application notes*]**.

FIA_USB.1.3 — The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: **[*rules defined in FMT_MSA.1.1/JCRE*].**

*Application Note: For S.PACKAGEs, the Context security attribute plays the role of the appropriate user security attribute; see **FMT_MSA.1.1/JCRE** above.*

6.2.5 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.

6.2.5.1 FPT_AMT.1 Abstract Machine Testing

Editorial changes: FPT_AMT.1 in [4] was removed according to Common Criteria 3.1 [13].

6.2.5.2 FPT_FLS.1 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 P5CD080V0B, P5CC080V0B that is conformant to [4].

FPT_FLS.1.1/SCP — The TSF shall preserve a secure state when the following types of failures occur: **[*exposure to operating conditions which may not be tolerated according to the requirement Limited fault tolerance (FRU_FLT.2) and where therefore a malfunction could occur and failures detected by TSF according to FPT_TST.1*].**

6.2.5.3 FRU_FLT.2/SCP Limited Fault Tolerance

The functional requirement FRU_FLT.2 is hierarchical to the requirement FRU_FLT.1 that is included in [1] and therefore includes this requirement. It has been taken over from the ST of the certified hardware platform P5CD080V0B, P5CC080V0B that is conformant to [4].

FRU_FLT.2.1/SCP — The TSF shall ensure the operation of all the TOE[27] capabilities when the following failures occur: **[*exposure to operating conditions which may not be tolerated according to the requirement Failure with preservation of a secure state (FPT_FLS.1)*].**

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

These components shall be used to specify the list of *SCP* capabilities supporting the Java Card System/CM that will still be operational at the occurrence of the mentioned failures (EEPROM worn out, lack of EEPROM, random generator failure).

6.2.5.4 FPT_PHP.3/SCP Resistance to Physical Attack

This functional requirement has been taken over from the ST of the certified hardware platform P5CD080V0B, P5CC080V0B that is conformant to [4].

FPT_PHP.3.1/SCP — The TSF shall resist **[*physical manipulation and physical probing*]** to the **[*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.

Editorial changes: The phrase “TSP is not violated” in [4] was replaced by “SFRs are always enforced” according to Common Criteria 3.1 [13].

[27] Editorial Change: The word “TOE’s” was replaced by “TOE”.

6.2.5.5 FDP_ACC.1/SCP Subset access control

This functional requirement has been taken over from the ST of the certified hardware platform P5CD080V0B, P5CC080V0B that is conformant to [4].

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.

6.2.5.6 FDP_ACF.1/SCP Security attribute based access control

This functional requirement has been taken over from the ST of the certified hardware platform P5CD080V0B, P5CC080V0B that is conformant to [4].

FDP_ACF.1.1/SCP — FDP_ACF.1.1 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	The TSF shall explicitly authorise 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	The TSF shall explicitly deny access of subjects to objects based on the following additional rules: [***assignment: none***].

6.2.5.7 FDP_RIP.1/SCP Subset information flow control

This functional requirement has been taken over from the ST of the certified hardware platform P5CD080V0B, P5CC080V0B that is conformant to [4].

FDP_RIP.1.1/SCP	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**:** [***assignment: all objects (variables) used by the Crypto Library as specified in the user guidance documentation.***]

Application Note: The TSF ensures that, upon exit from each function, with the exception of input parameters, return values or

locations where it is explicitly documented that values remain at specific addresses, any memory resources used by that function that contained temporary or secret values are cleared.

6.2.5.8 FDP_IFC.1 Subset information flow control

This functional requirement has been taken over from the ST of the certified hardware platform P5CD080V0B, P5CC080V0B that is conformant to [4].

FDP_IFC.1.1/SCP	The TSF shall enforce the **[*assignment: Data Processing Policy*]** on **[*assignment: all confidential data when they are processed or transferred by the TOE or by the Security IC Embedded Software*]**.

6.2.5.9 FDP_ITT.1/SCP Basic internal transfer protection

This functional requirement has been taken over from the ST of the certified hardware platform P5CD080V0B, P5CC080V0B that is conformant to [4].

FDP_ITT.1.1/SCP	The TSF shall enforce the **[*assignment: Data Processing Policy*]** to prevent the **[*selection: disclosure*]** of user data when it is transmitted between physically-separated parts of the TOE.
Refinement:	The different memories, the CPU and other functional units of the TOE (e.g. a cryptographic co-processor) are seen as physically-separated parts of the TOE.

6.2.5.10 FPT_ITT.1/SCP Basic internal TSF data transfer protection

This functional requirement has been taken over from the ST of the certified hardware platform P5CD080V0B, P5CC080V0B that is conformant to [4].

FPT_ITT.1.1/SCP	The TSF shall protect TSF data from **[*selection: disclosure*]** when it is transmitted between separate parts of the TOE.
Refinement:	The different memories, the CPU and other functional units of the TOE (e.g. a cryptographic co-processor) are seen as separated parts of the TOE.

6.2.5.11 FPT_SEP.1 TSF Domain Separation

Editorial changes: FPT_SEP.1.1/SCP and FPT_SEP.1.2/SCP in [4] were removed according to Common Criteria 3.1 [13].

6.2.5.12 FPT_RCV/SCP Trusted Recovery

FPT_RCV.3.1/SCP	When automated recovery from **[*detected integrity errors in D.APP_CODE, D.APP_I_DATA, D.PIN, D.APP_KEYs*]** is not possible, the TSF shall enter a maintenance mode where the ability to return the TOE to a secure state is provided.
FPT_RCV.3.2/SCP	For **[*power failure*]**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/SCP	The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **[*100%*]** for loss of TSF data or objects under the control of the TSF. *Editorial changes: The phrase "objects within the TSC." in [4] was replaced by "objects under the control of the TSF" according to Common Criteria 3.1 [13].*
FPT_RCV.3.4/SCP	The TSF shall provide the capability to determine the objects that were or were not capable of being recovered.
FPT_RCV.4.1/SCP	The TSF shall ensure that ***reading from and writing to static and objects' fields interrupted by power loss*** have the property that the function either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state. *Application Note: This requirement comes from the specification of the Java Card platform but is obviously supported in the implementation by a low-level mechanism.* *Application Note: In case of detected integrity errors in D.APP_CODE, D.APP_I_DATA, D.PIN, D.APP_KEYs, the card should stop and wait for the maintenance action reset. When the card is reset, the hardware should be re-initialized with the ability to return the TOE to a secure state (FPT_RCV.3.1/SCP).* *A power failure should cause a reset and after a reset a secure state must be entered (FPT_RCV.3.2/SCP).* *A transaction mechanism should ensure that no TSF data or objects are lost when a secure state is restored. Therefore, the TOE must check in advance if enough space is left in the transaction buffer (FPT_RCV.3.3/SCP, FPT_RCV.4.1/SCP).* *The capability to determine the objects that were or were not capable of being recovered (FPT_RCV.3.4/SCP) should be given by the kind of memory used for the objects* *Editorial changes: The word "SF" in [4] was replaced by "function" according to Common Criteria 3.1 [13].*

6.2.5.13 FPT_RVM.1 Reference Mediation

Editorial changes: FPT_RVM.1 in [4] was removed according to Common Criteria 3.1 [13].

6.2.5.14 FMT_MSA.3/SCP Static attribute initialisation

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 can not 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.

6.2.6 CMGRG Security Functional Requirements

This group contains the security requirements for the card manager. For this evaluation the card manager belongs to the TOE and the functional requirements are stated here as functional requirements for the TOE.

6.2.6.1 FDP_ACC.1/CMGR Subset Access Control

FDP_ACC.1.1/CMGR The TSF shall enforce the ***CARD CONTENT MANAGEMENT access control SFP*** on **[*subjects: S.PACKAGE(CM), S.PACKAGE, S.JCRE; objects: D.App_Code, and all operations among subjects and objects covered by the SFP*]**.

6.2.6.2 FDP_ACF.1/CMGR Security Attribute based Access Control

FDP_ACF.1.1/CMGR The TSF shall enforce the ***CARD CONTENT MANAGEMENT access control SFP*** to objects based on **[*the security attributes of S.PACKAGE(CM): Card Life Cycle State as defined in [GP] section 5.1: OP_READY, INITIALIZED, SECURED, CARD_LOCKED, TERMINATED*]**.

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

1. ***Loading of D.App_Code must only be performed by S.PACKAGE(CM) in card life cycle states OP_READY, INITIALIZED or SECURED and must not be performed in card life cycle states CARD_LOCKED or TERMINATED***
2. ***Loading of D.App_Code must only be performed S.PACKAGE(CM) after initiation of a Secure Channel.***
3. ***S.PACKAGE(CM) is allowed to set the Card Life Cycle States FUSED, PROTECTED, OP_READY, INITIALIZED, SECURED, CARD_LOCKED, and TERMINATED.*** **].**

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

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

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

6.2.6.3 FMT_MSA.1/CMGR Management of Security Attributes

FMT_MSA.1.1/CMGR	The TSF shall enforce the ***CARD CONTENT MANAGEMENT access control SFP*** to restrict the ability to **[*modify*]** the security attributes **[*card life cycle state*]** to **[*S.PACKAGE(CM)*]**.

6.2.6.4 FMT_MSA.3/CMGR Static Attribute Initialization

FMT_MSA.3.1/CMGR	The TSF shall enforce the ***CARD CONTENT MANAGEMENT access control SFP*** to provide ***restrictive*** default values for security attributes that are used to enforce the SFP.
FMT_MSA.3.2/CMGR	The TSF shall allow the **[*no roles*]** to specify alternative initial values to override the default values when an object or information is created.

6.2.6.5 FMT_SMR.1/CMGR Security Roles

FMT_SMR.1.1/CMGR	The TSF shall maintain the roles: **[*S.PACKAGE(CM)*]**.
FMT_SMR.1.2/CMGR	The TSF shall be able to associate users with roles.

6.2.6.6 FIA_UID.1/CMGR Timing of Identification

FIA_UID.1.1/CMGR	The TSF shall allow **[*execution of S.PACKAGE*]** on behalf of the user to be performed before the user is identified.
FIA_UID.1.2/CMGR	The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

6.2.7 Further Functional Requirements not contained in [1]

The SFR in this section are not contained in [JCSPP]. SFRs defined in sections 6.2.7.1 - 5.1.7.15 are specifically related to security functionality of the Card Manager.

6.2.7.1 FDP_ETC.1 Export of User Data without Security Attributes

FDP_ETC.1.1	The TSF shall enforce the **[*CARD CONTENT MANAGEMENT access control SFP*]** when exporting user data, controlled under the SFP(s), outside of the TOE.
FDP_ETC.1.2	The TSF shall export the user data without the user data's associated security attributes. *Editorial changes: The word "TSC" in [4] was replaced by "TOE" according to Common Criteria 3.1 [13].*

6.2.7.2 FDP_ITC.1 Import of User Data without Security Attributes

FDP_ITC.1.1	The TSF shall enforce the **[*CARD CONTENT MANAGEMENT access control SFP*]** when importing user data, controlled under the SFP, from outside of the TOE. *Application note: User data are: D.APP_CODE, D.PIN, D.APP_C_DATA, D.APP_I_DATA, D.APP_KEYs. The most common importation of user data is normally package loading and applet installation on the behalf of the installer. In the case of this ST,*

loading is handled separately during manufacturing and not through the card manager loader. Security attributes consist of the shareable flag of the class component, AID and version numbers of the package, maximal operand stack size and number of local variables for each method, and export and import components. Other instances of importing user data include setting the pin and key import.

FDP_ITC.1.2	The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.
FDP_ITC.1.3	The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC: **[none]**. *Editorial changes: The word "TSC" in [4] was replaced by "TOE" according to Common Criteria 3.1 [13].*

6.2.7.3 FIA_AFL.1 Basic authentication Failure Handling

FIA_AFL.1.1/PIN	The TSF shall detect when **[*an administrator configurable positive integer within [1 and 127]*]** unsuccessful authentication attempts occur related to **[*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 **[*block the authentication with D.PIN*].**
FIA_AFL.1.1/CMGR	The TSF shall detect when **[*66 consecutive*]** unsuccessful authentication attempts occur related to **[*any user authentication to the CARDMANAGER (S.PACKAGE(CM) via Secure Messaging using D.APP_KEYs*]**.
FIA_AFL.1.2/CMGR	When the defined number of unsuccessful authentication attempts has been ***surpassed***, the TSF shall **[*block the CARD*]**.

6.2.7.4 FIA_UAU.1 Timing of Authentication

FIA_UAU.1.1	The TSF shall allow **[*the following TSF mediated command*]** on behalf of the user to be performed before the user is authenticated.

Table 25. TSF mediated commands for FIA_UAU.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

FIA_UAU.1.2	The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

6.2.7.5 FIA_UAU.3/CMGR Unforgeable Authentication

FIA_UAU.3.1/CMGR	The TSF shall **[*prevent*]** use of authentication data that has been forged by any user of the TSF.
FIA_UAU.3.2/CMGR	The TSF shall **[*prevent*]** use of authentication data that has been copied from any other user of the TSF.
Note:	Only applicable for card manager authentication and not for authentication with D.PIN.

6.2.7.6 FIA_UAU.4/CMGR Single-use Authentication Mechanisms

FIA_UAU.4.1/CMGR	The TSF shall prevent reuse of authentication data related to **[*Card Manager authentication mechanism*]**.

6.2.7.7 FTP_ITC.1/CMGR Inter-TSF Trusted Channel – none

FTP_ITC.1.1/CMGR	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. *Editorial changes: The word "a remote" was replaced by "another" according to Common Criteria 3.1 [13].*
FTP_ITC.1.2/CMGR	The TSF shall permit **[*another trusted IT product*]** to initiate communication via the trusted channel.
FTP_ITC.1.3/CMGR	The TSF shall initiate communication via the trusted channel for **[*loading of D.App_Code, setting the Card Life Cycle State*]**.

6.2.7.8 FAU_SAA.1 Potential Violation Analysis

FAU_SAA.1.1	The TSF shall be able to apply a set of rules in monitoring the audited events and based upon these rules indicate a potential violation of the enforcement of the SFRs. *Editorial changes: The phrase "of the TSP" was replaced by "enforcement of the SFRs." according to Common Criteria 3.1 [13].*
FAU_SAA.1.2	The TSF shall enforce the following rules for monitoring audited events: a) Accumulation or combination of **[the following auditable events]** known to indicate a potential security violation; ***List of auditable events:***

1. Abnormal environmental conditions (frequency, voltage, temperature)
2. Physical tampering
3. EEPROM failure audited through exceptions in the read/write operations and inconsistency check;
4. Card Manager life cycle state inconsistency audited through the life cycle checks in all administrative operations and the self test mechanism on start-up.

5. Applet life cycle inconsistency.
6. Corruption of check-summed objects.
7. Illegal access to the previously defined D.JAVA_OBJECT objects audited through the firewall mechanism.
8. Unavailability of resources audited through the object allocation mechanism.
9. Abortion of a transaction in an unexpected context (see abortTransaction(),[7] and [8], §7.6.2)
10. Violation of the Firewall or JCVM SFPs.
11. Array overflow
12. Other runtime errors related to applet's failure (like uncaught exceptions)
13. Card tearing (unexpected removal of the Card out of the CAD) and power failure

b) [*no other rules*].

Application Note: *Off-card entities are provided with the basic ability to find out certain pieces of information about the card through the Open Platform GET DATA command. Authenticated off-card entities can determine other information such as the AIDs of on-card Applications and Life Cycle states using other APDU commands. This provides a limited ability to "audit" the card, and is probably sufficient for most purposes.*

6.2.7.9 FAU_SAS.1/SCP Audit Data Storage

FAU_SAS.1.1/SCP	The TSF shall provide **[*test personnel before TOE Delivery*]** with the capability to store the **[*Initialisation Data and/or Prepersonalisation Data and/or supplements of the Smartcard Embedded Software*]** in the ***[assignment: audit records]***.

Remark: The word "the test process" in BSI- PP-0035 [27] was replaced by "test personel" according to BSI- PP-0002 [4].

6.2.7.10 FMT_SMF.1 Specification of Management Function

FMT_SMF.1.1	The TSF shall be capable of performing the following management functions: **[*modify the behavior of functions, modify the active context and the SELECTed applet Context, modify the list of registered applets' AID, modify the card life cycle state attribute*]**.

6.2.7.11 FCS_RNG.1 Quality metric for Random Numbers

FCS_RNG.1.1	The TSF shall provide a ***hybrid*** random number generator that implements: ***a deterministic RNG according to ANSI X9.31. It is seeded with random numbers from the physical RNG of the hardware. The physical RNG of the hardware is tested according to the Guidance Manual of the hardware before it is used to derive the seed for the deterministic RNG. This happens at startup of the operating system. In case of failure of the physical RNG the card is muted.***

FCS_RNG.1.2	The TSF shall provide random numbers that meet ***class K3 of [21]***.

6.2.7.12 FPT_EMSEC.1 TOE Emanation

FPT_EMSEC.1.1	The TOE shall not emit ***variations in power consumption or timing during command execution*** in excess of ***non-useful information*** enabling access to **[*TSF data: D.JCS_KEYs and D.CRYPTO*]** and **[*User data: D.PIN, D.APP_KEYs*].**
FPT_EMSEC.1.2	The TSF shall ensure ***that unauthorized users*** are unable to use the following interface **electrical contacts** to gain access to **[*TSF data: D.JCS_KEYs and D.CRYPTO*]** and **[*User data: D.PIN, D.APP_KEYs*]**.

6.2.7.13 FMT_LIM.1 Limited Capabilities

FMT_LIM.1.1	The TSF shall be designed in a manner that limits their capabilities so that in conjunction with “Limited availability (FMT_LIM.2)” the following policy is enforced ***Deploying Test Features after TOE Delivery does not allow***

1. ***User Data to be disclosed or manipulated***
2. ***TSF data to be disclosed or manipulated***
3. ***software to be reconstructed and***
4. ***substantial information about construction of TSF to be gathered which may enable other attacks*.**

6.2.7.14 FMT_LIM.2 Limited Availability

FMT_LIM.2.1	The TSF shall be designed in a manner that limits their availability so that in conjunction with “Limited capabilities (FMT_LIM.1)” the following policy is enforced ***Deploying Test Features after TOE Delivery does not allow***

1. ***User Data to be disclosed or manipulated***
2. ***TSF data to be disclosed or manipulated***
3. ***software to be reconstructed and***
4. ***substantial information about construction of TSF to be gathered which may enable other attacks.***

6.2.7.15 FPT_PHP.1 Passive Detection of physical Attack

FPT_PHP.1.1	The TSF shall provide unambiguous detection of physical tampering that might compromise the TSF.
FPT_PHP.1.2	The TSF shall provide the capability to determine whether physical tampering with the TSF's devices or TSF's elements has occurred.

6.2.7.16 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 (shared between the card manager and the TOE), the bytecode and its data arguments (shared with***

	applets and API packages), when shared between the TSF and another trusted IT product.
FPT_TDC.1.2	The TSF shall use ***the following rules*** when interpreting the TSF data from another trusted IT product: • ***The Java Card Virtual Machine specification [9];*** • ***Reference export files;*** • ***The ISO 7816-6 rules;*** • ***The EMV specification*** *Editorial changes: List moved to the end of the SFR.*

6.3 Security assurance requirements (SARs)

The assurance requirements of this evaluation are EAL5 augmented by ALC_DVS.2 and AVA_VAN.5.

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.5** Complete semi-formal functional specification with additional error information
- **ADV_ARC.1** Security architecture description
- **ADV_TDS.4** Semiformal modular design
- **ADV_INT.2** Well-structured internals

These SARs ensure that the TOE will be able to meet its security requirements and fulfill its objectives. The *Java Card System* shall implement the [7]. The implementation of the Java Card API shall be designed in a secure manner, including specific techniques to render sensitive operations resistant to state-of-art attacks.

- **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 *package*s programmers should have a complete understanding of the concepts defined in [8] and [9]. 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 *applet*s (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.5** Development tools CM coverage

These components contribute to the integrity and correctness of the TOE during its development. Procedures dealing with physical, personnel, organizational, technical measures for the confidentiality and integrity of *Java Card System* software (source code and any associated documents) shall exist and be applied in software development.

- **ALC_LCD.1** Developer defined life-cycle model
- **ALC_TAT.2** 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.3** 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 EAL5 results from the selection of the following two SARs:

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

EAL5 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.

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

EAL5 requires for the vulnerability assessment the assurance component **AVA_VAN.4**. 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, EAL5 is augmented with the component **AVA_VAN.5**, which requires that the TOE is resistant against attackers processing high attack potential.

The refinements given in (BSI-PP-0002) Smartcard IC Platform Protection profile [4] are not applied for this TOE because of newer assurance components in Common Criteria 3.1 [15].

6.4 SARs and the security requirement rationale

6.4.1 Security Functional Requirements Rationale

This section proves that the quantity of security requirements (TOE and environment) is suited to fulfill the security objectives described in section 4 and that it can be traced back to the security objectives.

6.4.1.1 TOE Security Requirements Rationale

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 26. Assignment: TOE security requirements – TOE security objectives

	O.PROTECT_DATA	O.SIDE_CHANNEL	O.OS_DECEIVE	O.FAULT_PROTECT	O.PHYSICAL	O.IDENTIFICATION	O.CARD-MANAGEMENT	O.SHRD_VAR_INTEG	O.SHRD_VAR_CONFID	O.FIREWALL	O.OPERATE	O.ALARM	O.RESOURCES	O.REALLOCATION	O.SID	O.MF_FW	O.SCP.IC	O.SCP.RECOVERY	O.SCP.SUPPORT	O.CIPHER	O.PIN-MNGT	O.KEY-MNGT	O.TRANSACTION	O.RND
FAU																								
FAU_ARP.1/ JCS											x	x	x											
FAU_SAA.1			x	x							x													
FAU_SAS.1						x																		
FCS																								
FCS_CKM.1																			x		x			
FCS_CKM.2																			x		x			
FCS_CKM.3	x	x																	x		x			
FCS_CKM.4	x	x																	x		x			
FCS_COP.1 (all iterations)	x	x																	x		x			
FCS_RNG.1																								x

	O.PROTECT_DATA	O.SIDE_CHANNEL	O.OS_DECEIVE	O.FAULT_PROTECT	O.PHYSICAL	O.IDENTIFICATION	O.CARD-MANAGEMENT	O.SHRD_VAR_INTEG	O.SHRD_VAR_CONFID	O.FIREWALL	O.OPERATE	O.ALARM	O.RESOURCES	O.REALLOCATION	O.SID	O.MF_FW	O.SCP.IC	O.SCP.RECOVERY	O.SCP.SUPPORT	O.CIPHER	O.PIN-MNGT	O.KEY-MNGT	O.TRANSACTION	O.RND
FDP																								
FDP_ACC.1/ CMGR							x																	
FDP_ACC.1/ SCP																x								
FDP_ACC.2/ FIREWALL	x									x	x										x			
FDP_ACF.1 /FIREWALL	x									x	x										x			
FDP_ACF.1/ CMGR							x																	
FDP_ACF.1/ SCP																x								
FDP_ETC.1	x										x													
FDP_IFC.1/ JCVM								x	x	x														
FDP_IFC.1/ SCP		x		x							x									x				
FDP_IFF.1/ JCVM								x	x	x														
FDP_ITC.1	x										x													
FDP_ITT.1/ SCP		x		x							X									x				
FDP_RIP.1 (all iterations)	x								x		x			x							x	x	x	
FDP_ROL.1/ FIREWALL																					x		x	
FDP_SDI.2	x										x										x	x		
FIA																								
FIA_AFL.1/PIN	x										x													
FIA_AFL.1/ CMGR							x																	

	O.PROTECT_DATA	O.SIDE_CHANNEL	O.OS_DECEIVE	O.FAULT_PROTECT	O.PHYSICAL	O.IDENTIFICATION	O.CARD-MANAGEMENT	O.SHRD_VAR_INTEG	O.SHRD_VAR_CONFID	O.FIREWALL	O.OPERATE	O.ALARM	O.RESOURCES	O.REALLOCATION	O.SID	O.MF_FW	O.SCP.IC	O.SCP.RECOVERY	O.SCP.SUPPORT	O.CIPHER	O.PIN-MNGT	O.KEY-MNGT	O.TRANSACTION	O.RND
FIA_ATD.1/AID	x										x				x									
FIA_UAU.1	x										x													
FIA_UAU.3/ CMGR	x										x													
FIA_UAU.4/ CMGR	x										x													
FIA_UID.1/ CMGR	x						x				x													
FIA_UID.2/AID															x									
FIA_USB.1	x										x				x									
FMT																								
FMT_LIM.1	x																							
FMT_LIM.2	x																							
FMT_MSA.1/ JCRE			x							x					x									
FMT_MSA.1/ CMGR							x																	
FMT_MSA.2/ JCRE			x							x														
FMT_MSA.3/ FIREWALL			x							x					x									
FMT_MSA.3/ CMGR							x																	
FMT_MSA.3/ SCP																x								
FMT_MTD.1/ JCRE	x		x							x			x		x									
FMT_MTD.3										x			x		x									
FMT_SMF.1	x		x				x			x	x		x											
FMT_SMR.1/ JCRE	x									x	x		x											
FMT_SMR.1/							x																	

	O.PROTECT_DATA	O.SIDE_CHANNEL	O.OS_DECEIVE	O.FAULT_PROTECT	O.PHYSICAL	O.IDENTIFICATION	O.CARD-MANAGEMENT	O.SHRD_VAR_INTEG	O.SHRD_VAR_CONFID	O.FIREWALL	O.OPERATE	O.ALARM	O.RESOURCES	O.REALLOCATION	O.SID	O.MF_FW	O.SCP.IC	O.SCP.RECOVERY	O.SCP.SUPPORT	O.CIPHER	O.PIN-MNGT	O.KEY-MNGT	O.TRANSACTION	O.RND
CMGR																								
FPR																								
FPR_UNO.1		x									x									x	x	x		
FPT																								
FPT_EMSEC.1		x																						
FPT_FLS.1/ JCS											x	x	x					x						
FPT_FLS.1/ SCP				x							x	x	x					x		x				
FPT_ITT.1/ SCP		x		x																x				
FPT_PHP.1					x												x							
FPT_PHP.3/ SCP					x												x							
FPT_RCV.3/ SCP																		x	x					
FPT_RCV.4/ SCP																			x					
FPT_TDC.1											x													
FPT_TST.1				x							x													
FRU																								
FRU_FLT.2/ SCP				x														x		x				
FTP																								
FTP_ITC.1/ CMGR							x																	

The explanations given in the full Security Target were not intended to be published and have therefore been removed from this Security Target Lite.

6.4.2 SARs rationale

6.4.2.1 Evaluation Assurance Level Rationale

An assurance requirement of **EAL5** 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. **EAL5** represents the highest practical level of assurance expected for a commercial grade product.

In order to provide a meaningful level of assurance that the TOE provides an adequate level of defense against such attacks, the evaluators should have access to the low level design and source code. The lowest for which such access is required is **EAL5**.

The assurance level **EAL5** is achievable, since it requires no specialist techniques on the part of the developer.

6.4.2.2 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 EAL5 (only ALC_DVS.1 is found in EAL5). 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 EAL5 (only AVA_VAN.4 is found in EAL5).

AVA_VAN.4 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 EAL5, and so these dependencies are satisfied.

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 27. List of all security functions

No	TOE Security Function	Short Description
1.	SF.AccessControl	enforces the access control
2.	SF.Audit	Audit functionality
3.	SF.CryptoKey	Cryptographic key management
4.	SF.CryptoOperation	Cryptographic operation
5.	SF.I&A	Identification and authentication
6.	SF.SecureManagement	Secure management of TOE resources
7.	SF.PIN	PIN management
8.	SF.Transaction	Transaction management
9.	SF.Hardware	TSF of the underlying IC
10.	SF.CryptoLib	TSF of the certified crypto library

7.1.1 SF.AccessControl

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

1 CARD CONTENT MANAGEMENT access control SFP (see sections 6.2.6.1 *FDP_ACC.1/CMGR* and 6.2.6.2 *FDP_ACF.1/CMGR*) for the import and export of user data (see sections 6.2.7.1 *FDP_ETC.1*, 6.2.7.2 *FDP_ITC.1*), loading of applet and library code (D.App_Code) and setting the card life cycle state via a trusted channel (see section 6.2.7.7 *FTP_ITC.1/CMGR*).

2 FIREWALL access control SFP (see sections 6.2.1.1 *FDP_ACC.2/FIREWALL* and 6.2.1.2 *FDP_ACF.1/FIREWALL*), and

3 JCVM information flow control SFP (see section 6.2.1.3 *FDP_IFC.1/JCVM*).

It further ensures the management of the necessary security attributes:

4 Only S.PACKAGE(CM) is allowed to modify the card life cycle state (see sections 6.2.1.6 *FMT_MSA.1/CMGR*, 6.2.7.10 *FMT_SMF.1*, and 6.2.1.9 *FMT_SMR.1/CMGR)*.

5 Only the JCRE (S.JCRE) can modify the active context and the SELECTed applet Context security attributes and can change the list of registered applets' AID (see 6.2.6.3 *FMT_MSA.1/JCRE*, 6.2.4.1 *FMT_MTD.1/JCRE*, 6.2.7.10 *FMT_SMF.1*, and *FMT_SMR.1/JCRE*.

6 Only secure values are accepted for TSF data and security attributes (see 6.2.1.7 *FMT_MSA.2/JCRE*, 6.2.4.2 *FMT_MTD.3*, 6.2.7.10 *FMT_SMF.1*, 6.2.1.9 *FMT_SMR.1/CMGR*, *FMT_SMR.1/JCRE)*. 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.

7 Restrictive default values are used for the security attributes, which cannot be overwritten (see 6.2.1.8 *FMT_MSA.3/CMGR* and *FMT_MSA.3/FIREWALL)*

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 (see 6.2.7.8):

1. Abnormal environmental conditions (frequency, voltage, temperature), in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1*, and *FPT_FLS.1/JCS*.
2. Physical tampering, in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1*, *FPT_FLS.1/JCS, FPT_PHP.1* and *FPT_PHP.3/SCP*.
3. EEPROM failure audited through exceptions in the read/write operations and consistency/integrity check, in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1* and *FPT_FLS.1/JCS*.
4. Card Manager 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/JCS*, *FAU_SAA.1* and *FPT_FLS.1/JCS*.
5. Applet life cycle inconsistency, in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1* and *FPT_FLS.1/JCS*.
6. Corruption of check-summed objects, in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1* and *FPT_FLS.1/JCS*.
7. Illegal access to the previously defined D.JAVA_OBJECT objects audited through the firewall mechanism, in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1,* and *FPT_FLS.1/JCS*.
8. Unavailability of resources audited through the object allocation mechanism, in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1* and *FPT_FLS.1/JCS*
9. Abortion of a transaction in an unexpected context (see [7] and [8], §7.6.2), in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1* and *FPT_FLS.1/JCS*.

Based on the events listed above and the following events (also see 6.2.3.1):

10. Violation of the Firewall or JCVM SFPs, in fulfillment of *FAU_ARP.1/JCS, FAU_SAA.1*, and *FPT_FLS.1/JCS*.
11. Array overflow, in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1* and *FPT_FLS.1/JCS*.
12. Other runtime errors related to applet's failure (like uncaught exceptions), in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1* and *FPT_FLS.1/JCS*.
13. Card tearing (unexpected removal of the Card out of the CAD) and power failure, in fulfillment of *FAU_ARP.1/JCS*, *FAU_SAA.1* and *FPT_FLS.1/JCS*.

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 6.2.5.4) according to the ST lite **[5]** or **[22]**.

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 (**[5]**/[19] chapter 6.1] 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 28. Response/Reaction on SF.Audit events

Event #	Exception	Terminate card	HW Reset IC or other HW action	Lock card session
1. Abnormal environmental conditions			X	
2. Physical tampering			X	X
3.1 EEPROM failure audited through exceptions in the read/write operations and consistency/integrity check				X
3.2 self test mechanism on start-up				X
4. Card Manager life cycle state inconsistency audited through the life cycle checks in all administrative operations		X		
5. Applet life cycle inconsistency		X		
6. Corruption of check-summed objects				X
7. Illegal access to the previously defined D.JAVA_OBJECT objects audited through	X			X

Event #	Exception	Terminate card	HW Reset IC or other HW action	Lock card session
the firewall mechanism.				
8. Unavailability of resources audited through the object allocation mechanism.	X			
9. Abortion of a transaction in an unexpected context	X			
10. Violation of the Firewall or JCVM SFPs	X			
11. Array overflow	X			
12. Other runtime errors related to applet's failure (like uncaught exceptions)	X			
13. 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:

1. Generation of DES keys with length of 112 and 168 Bit based on random numbers according to AIS 20 [21] class K3 (see 6.2.2.1 *FCS_CKM.1*).
2. Generation of RSA keys with length from 1976 to 2048 Bit based on random numbers according to AIS 20 [21] class K3 (see 6.2.2.1 *FCS_CKM.1*).
3. Generation of AES keys with length of 128, 192, and 256 Bit based on random numbers according to AIS 20 [21] class K3 (see 6.2.2.1 *FCS_CKM.1*).
4. Distribution of DES keys with the method *setKey* of Java Card API [7] (see 6.2.2.2 *FCS_CKM.2*).
5. Distribution of RSA keys with the method *setExponent* and *setModulus* of Java Card API [7] (see 6.2.2.2 *FCS_CKM.2*).
6. Distribution of AES keys with the method *setKey* of Java Card API [7] (see 6.2.2.2 *FCS_CKM.2*).
7. Management of DES, AES, EC, and RSA- keys with methods/commands defined in packages javacard.security and javacardx.crypto of Java Card API [7] (see 6.2.2.3 *FCS_CKM.3*).
8. Destruction of DES, AES, EC, and RSA- keys by physically overwriting the keys by method clearKey of Java Card API [7] (see 6.2.2.4 *FCS_CKM.4*).
9. Generation of ECC over GF(p) keys with length from 224 to 320 Bit based on random numbers according to AIS 20 [21] class K3 (see 6.2.2.1 *FCS_CKM.1*).
10. Distribution of ECC over GF(p) keys with the method *setA, setB, setFieldFP, setG, setK, and setR* of Java Card API [7] (see 6.2.2.2 *FCS_CKM.2*).

11. Destruction of session keys by physically overwriting the keys by overwriting them with zeros when explicitly deleted or when the applet is deselected (see 6.2.2.4 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:

1. Data encryption and decryption with Triple-DES in ECB/CBC Mode and cryptographic key sizes of 112 and 168 Bit that meets FIPS 46-3 [20] (see 6.2.2.5 *FCS_COP.1/TripleDES*)
2. Data encryption and decryption with RSA and PKCS#1 padding [16]. Key sizes range from 1976 to 2048 Bit (see 6.2.2.5 *FCS_COP.1/ RSACipher*).
3. 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 [24] (see 6.2.2.5 *FCS_COP.1/ DESMAC*).
4. Data encryption and decryption with AES in ECB/CBC Mode and cryptographic key sizes of 128, 192, and 256 Bit that meets FIPS 197 [19] (see 6.2.2.5 *FCS_COP.1/ AES*).
5. RSA digital signature generation and verification with SHA-1 as hash function and cryptographic key sizes from 1976 to 2048 Bit according to ISO 9796-2 [23] (see 6.2.2.5 *FCS_COP.1/RSASignatureISO9796*).
6. RSA digital signature generation and verification with SHA-1 as hash function and cryptographic key sizes from 1976 to 2048 Bit according to PKCS#1 [16] (see 6.2.2.5 *FCS_COP.1/ RSASignaturePKCS#1*).
7. Secure hash computation with SHA-1 according to FIPS 180-3 [18] (see 6.2.2.5 *FCS_COP.1/ SHA-1*).
8. Random number generation according to AIS 20 [21] class K3 (see 6.2.7.11 *FCS_RNG.1*).
9. EC digital signature generation and verification with SHA-1, SHA-224, and SHA-256 as hash functions and cryptographic key sizes from 224 to 320 Bit according to ISO14888-3 [25] (see 6.2.2.5 FCS_COP.1/ECSignature).
10. Secure hash computation with SHA-224 according to FIPS 180-3 [18] (see 6.2.2.5 *FCS_COP.1/ SHA-224*).
11. Secure hash computation with SHA-256 according to FIPS 180-3 [18] (see 6.2.2.5 *FCS_COP.1/ SHA-256*).
12. Secure Messaging functionality for ICAO – encryption and decryption with Triple-DES in CBC mode and cryptographic key size of 112 bit FIPS 46-3 [20], as well as message authentication code with Retail MAC and cryptographic key size of 112 bit

according to ISO 9797-1 [24] (see 6.2.2.5 *FCS_COP.1/TDES_MRTD* and *FCS_COP.1/MAC_MRTD*).[28]

13. Diffie-Hellman key agreement with ECC over GF(p) and cryptographic key sizes from 224 to 320 bit according to ISO 11770-3 [26] (see 6.2.2.5 *FCS_COP.1/DHKeyExchange*).
14. Secure point addition in accordance with the specified cryptographic algorithm ECC over GF(p) and cryptographic key sizes 224 to 320 Bit according to ISO14888-3 [25] (see 6.2.2.5 FCS_COP.1/ECAdd)

7.1.5 SF.I&A

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

1. The TSF provide 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 (see 6.2.7.5 *FIA_UAU.3/CMGR* and 6.2.7.6 *FIA_UAU.4/CMGR*).
2. The TSF blocks the card when 66 consecutive unsuccessful card manager authentication attempts via secure messaging using D.APP_KEY occur (see 6.2.7.3 *FIA_AFL.1/CMGR*).
4. Package execution is possible before authentication (see 6.2.6.6 *FIA_UID.1/CMGR*).

7.1.6 SF.SecureManagment

The TSF provide a secure management of TOE resources:

5. 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 ([9], §6.5) (see 6.2.4.3 *FIA_ATD.1/AID*, 6.2.4.4 *FIA_UID.2/AID* and 6.2.4.5 *FIA_USB.1*).
6. The TSF run a suite of self-tests during initial start-up (at each power on) to demonstrate the correct operation of the TSF, to verify the integrity of TSF data, and to verify the integrity of stored TSF executable code. This includes checking the EEPROM integrity. If an error is detected, the TOE enters into a secure state (lock card session) (see 6.2.5.2 *FPT_FLS.1/SCP*, 6.2.5.4 *FPT_PHP.1* and 6.2.6.3 *FPT_TST.1*).
7. The TSF ensures that packages are unable to observe operations on secret keys and PIN codes by other subjects (see 6.2.3.5 *FPR_UNO.1*).
8. The TSF monitors user data D.APP_CODE, D.APP_I_DATA, D.PIN, D.APP_KEYs for integrity errors. If an error occurs, the TSF maintain a secure state (lock card session) (see 6.2.3.2 *FDP_SDI.2, 6.2.5.12 FPT_RCV.3/SCP*).
9. The TSF makes any previous information content of a resource unavailable upon (see 6.2.2.6 *FDP_RIP.1/OBJECTS*, *FDP_RIP.1/APDU*, *FDP_RIP.1/bArray*, *FDP_RIP.1/TRANSIENT*, *FDP_RIP.1/ABORT* and *FDP_RIP.1/KEYS*):
    - allocation of class instances, arrays, and the APDU buffer,

28. Other secure messaging functionality is part of the SF.CryptoOperation 1 and 3. Key destruction for ICAO functionality is part of SF.CryptoKey 11.

- de-allocation of bArray object, any transient object, any reference to an object instance created during an aborted transaction, and cryptographic buffer (D.CRYPTO).

10. 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.[29] All functions of SF.CryptoOperation except with SHA are resistant to side-channel attacks (e.g. timing attack, SPA, DPA, DFA, EMA, DEMA) (see 6.2.7.12 *FPT_EMSEC.1*).
11. CAP files, the bytecode and its data arguments are consistently interpreted using the following rules (see 6.2.7.16 *FPT_TDC.1*):
    a. The virtual machine specification [9];
    b. Reference export files;
    c. The ISO 7816-6 rules;
    d. The EMV specification.

7.1.7 SF.PIN

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

1. The TSF provide user authentication with a Global-PIN that is at least 6 digits long (see 6.2.7.4 *FIA_UAU.1*).
2. The maximum possible number of consecutive unsuccessful PIN-authentication attempts is user configurable number from 1 to 127. (see 6.2.7.3 *FIA_AFL.1/PIN*)
3. When this number has been met or surpassed, the PIN-authentication is blocked (see 6.2.7.3 *FIA_AFL.1/CMGR* and *FIA_AFL.1/PIN*).
4. Only the following commands are allowed, before successful authentication (see 6.2.7.4 *FIA_UAU.1*):
    - *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

[29] 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.

7.1.8 SF.Transaction

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 [8], §7.7, within the bounds of the Commit Capacity ([8], §7.8), and those described in Java Card API [7]. (see 6.2.2.7 FDP_ROL.1/FIREWALL, 2, 6.2.5.12 FPT_RCV.4/SCP).

7.1.9 SF.Hardware

The certified hardware (part of the TOE) features the following TSF. The exact formulation can be found in the hardware security target **[5]**:

1. Random Number Generator (F.RNG) used for SF.CryptoOperation 8 (see 6.2.7.11 *FCS_RNG.1*).
2. Triple-DES Co-processor (F.HW_DES) used for SF.CryptoOperation 1 and SF.CryptoLib 2 (see 6.2.2.5 *FCS_COP.1/TripleDES, 6.2.2.5 FCS_COP.1/TDES_MRTD and FCS_COP.1/MAC_MRTD*).
3. AES Co-processor (F.HW_AES) used for SF.CryptoOperation 4 (see 6.2.2.5 *FCS_COP.1/AES*).
4. Control of Operating Conditions (F.OPC) (see 6.2.5.2 *FPT_FLS.1/SCP*, 6.2.5.3 *FRU_FLT.2/SCP*).
5. Protection against Physical Manipulation (F.PHY) (see 6.2.2.5 *FCS_COP.1/TripleDES* and *FCS_COP.1/AES*, 6.2.7.11 *FCS_RNG.1*, 6.2.3.4 *FPT_FLS.1/SCP*, 6.2.7.15 *FPT_PHP.1*, 6.2.5.4 *FPT_PHP.3/SCP*, 6.2.5.3 *FRU_FLT.2/SCP,* 6.2.7.9 *FAU_SAS.1/SCP*, 6.2.5.8 *FDP_IFC.1/SCP*, 6.2.5.9 *FDP_ITT.1/SCP,* 6.2.5.10 *FPT_ITT.1/SCP*, *FMT_LIM.1*, 6.2.7.14 *FMT_LIM.2,* 6.2.5.5 *FDP_ACC.1/SCP*, 6.2.5.6 *FDP_ACF.1/SCP*, and 6.2.5.14 *FMT_MSA.3/SCP*).
6. Logical Protection (F.LOG) (see 6.2.7.12 *FPT_EMSEC.1*, 6.2.5.9 *FDP_ITT.1/SCP*, 6.2.5.10 *FPT_ITT.1/SCP*, and 6.2.5.8 *FDP_IFC.1/SCP*).
7. Protection of Mode Control (F.COMP) (see 6.2.7.13 *FMT_LIM.1*, 6.2.7.14 *FMT_LIM.2* and 6.2.7.9 *FAU_SAS.1/SCP*).
8. Memory Access Control (F.MEM_ACC). The functionality of the hardware is used for the MIFARE firewall (see 6.2.5.5 *FDP_ACC.1/SCP*, 6.2.5.6 *FDP_ACF.1/SCP*, and 6.2.5.14 *FMT_MSA.3/SCP*)
9. Special Function Register Access Control (F.SFR_ACC). The functionality of the hardware is not used by the TOE and not exposed at external interfaces of the composite TOE. Also the Crypto Library makes no use of the Special Function Register Access Control.

7.1.10 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 **[22]**:

1 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.

2 Software DES (F.DES) based on F.HW_DES used for SF.CryptoOperation 12 (see 6.2.2.5 FCS_COP.1/TDES_MRTD and FCS_COP.1/MAC_MRTD).

3 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.

4 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.

5 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.

6 ECC Signature Generation, Signature Verification (F.ECC_GF_p_ECDSA) used for SF.CryptoOperation 9 (see 6.2.2.5 FCS_COP.1/ECSignature)

7 Diffie-Hellman Key Exchange (F.ECC_GF_p_DH_KeyExch) used for SF.CryptoOperation 13 (see 6.2.2.5 FCS_COP.1/DHKeyExchange).

8 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.

9 EC Key Generation (F.ECC_GF_p_KeyGen) used for SF.CryptoKey 9 (see 6.2.2.1 FCS_CKM.1).

10 Compute the Secure Hash Algorithms (F.SHA) used for SF.CryptoOperation 7, 10, and 11 (see 6.2.2.5 FCS_COP.1/SHA-1, 6.2.2.5 FCS_COP.1/SHA-224, 6.2.2.5 FCS_COP.1/SHA-256)

11 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.

12 Clear memory areas used by the Crypto Library after usage (F.Object_Reuse) is used for SF.CryptoLib 2, 6, 7 and 9 (see 6.2.5.7 FDP_RIP.1/SCP).

13 Logical Protection (F.LOG) extends F.LOG of the Hardware and is used for SF.CryptoLib 2, 6, 7, 9, 10 and 16 (see 6.2.7.12 FPT_EMSEC.1, 6.2.5.9 FDP_ITT.1/SCP, 6.2.5.10 FPT_ITT.1/SCP, 6.2.5.8 FDP_IFC.1/SCP, and 6.2.5.2 FPT_FLS.1/SCP).

14 Cryptographic Key Destruction (FCS_CKM.4). The functionality of the cryptographic library is not used by the TOE and not exposed at external interfaces of the composite TOE.

15 ECC Point addition (FCS_COP.1[ECC_ADD] in F.ECC_GF_p_ECDSA) used for SF.CryptoOperation14 (see 6.2.2.5 FCS_COP.1/ECAdd).

16 Memory copy in a manner protected against side channel attacks (F.COPY)

8. Bibliography

[1] Java Card System - Minimal Configuration Protection Profile, Version 1.1, May 2006, part of: Java Card Protection Profile Collection, Version 1.1, May 2006

[2] Protection Profile - Smart Card Native Operating System Draft, Version 0.5, Issue April 2004

[3] 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)

[4] Smartcard IC Platform Protection Profile (SSVG-PP), Version 1.0, July 2001; registered and certified by (BSI) under the reference BSI-PP-0002-2001

[5] P5CD080/P5CN080/P5CC080/P5CC073V0B Security Target Lite, Rev 1.7, 28 September 2009, BSI-DSZ-CC-0410-2007

[6] Assurance Continuity Maintenance Report, NXP Secure Smart Card Controller P5CD080, P5CC080, P5CN080 and P5CC073V0B with specific IC Dedicated Software, 6 November 2009, BSI-DSZ-CC-0410-2007-MA-07

[7] Application Programming Interface, Java Card(tm) Platform, Version 2.2.2, March 2006, Sun Microsystems

[8] Runtime Environment Specification, Java Card(tm) Platform, Version 2.2.2, March 2006, Sun Microsystems

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

[10] Global Platform, Card Specification, Version 2.1.1, March 2003

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

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

[13] Common Criteria for Information Technology Security Evaluation, Part 1, Version 3.1, Revision 3, July 2009

[14] Common Criteria for Information Technology Security Evaluation, Part 2, Version 3.1, Revision 3, July 2009

[15] Common Criteria for Information Technology Security Evaluation, Part 3, Version 3.1, Revision 3, July 2009

[16] PKCS1: RSA Encryption Standard - An RSA Laboratories Technical Note, Version 1.5, Revised November 1, 1993

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

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

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

[20] 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

[21] Anwendungshinweise und Interpretationen zum Schema, AIS 20: Funktionalitaetsklassen und Evaluationsmethodologie fuer deterministische Zufallszahlengeneratoren, Version 1, 02.12.1999, Bundesamt fuer Sicherheit in der Informationstechnik

[22] Crypto Library V2.6 on P5CD080V0B / P5CN080V0B / P5CC080V0B / P5CC073V0B, Security Target Rev. 2.3, 12 November 2010, BSI-DSZ-CC-0709

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

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

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

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

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

[28] -

[29] P5Cx012/02x/040/073/080/144V0B family, Guidance, Delivery and Operation Manual, Rev. 1.8, Doc. ID 129918, 15 February 2010

[30] P5Cx012/02x/40/73/80/144 family Secure dual interface and contact PKI smart card controller, Rev. 3.7, Doc. ID 126537

[31] Bundesnetzagentur fuer Elektrizitaet, Gas, Telekommunikation, Post und Eisenbahnen - Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Uebersicht ueber geeignete Algorithmen), 06. Januar 2010

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

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

General — 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.

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 medical, military, aircraft, space or life support equipment, nor in applications where failure or malfunction of a 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 for 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.

10. Contents

1. ST Introduction (ASE_INT) 7
1.1 ST reference and TOE reference 7
1.2 TOE overview 7
1.3 TOE description 8
1.3.1 TOE abstract and definition 8
1.3.2 TOE Life-Cycle 11
1.3.3 Java Card Technology 13
1.3.4 Smart Card Platform 15
1.3.5 Native Applications 15
1.4 TOE Intended Usage 16
2. Conformance claims (ASE_CCL) 17
2.1 CC Conformance Claim 17
2.2 Statement of Compatibility concerning Composite Security Target 17
2.2.1 Separation of the Platform-TSF 17
2.2.2 Compatibility between the Composite Security Target and the Platform Security Target 22
3. Security problem definition (ASE_SPD) 28
3.1 Introduction 28
3.2 Assets 29
3.2.1 User Data 29
3.2.2 TSF Data 30
3.3 Threats 30
3.3.1 Threats not contained in [1] 31
3.3.1.1 Unauthorized full or partial Cloning of the TOE 32
3.3.1.2 Threats on TOE operational environment 32
3.3.1.3 Software Threats 32
3.3.1.4 Environment Threats on the complete TOE 34
3.3.1.5 Threat on Random Numbers 34
3.3.2 Threats from [1] 35
3.3.2.1 Confidentiality 35
3.3.2.2 Integrity 36
3.3.2.3 Identity Usurpation 36
3.3.2.4 Unauthorized Execution 37
3.3.2.5 Denial of Service 37
3.4 Organisational security policies (OSPs) 37
3.4.1 Organizational Security Policies from [1] 37
3.4.2 Additional Organizational Security Policies 37
3.5 Assumptions 37
3.5.1 Assumptions not contained in [1] 38
3.5.1.1 Assumption on Phase 7 38
3.5.2 Assumptions from [1] 38
3.6 Security Aspects 39
3.6.1 Confidentiality 39
3.6.2 Integrity 39
3.6.3 Unauthorized Executions 40
3.6.3.1 Bytecode Verification 40
3.6.3.2 CAP File Verification 41
3.6.3.3 Integrity and Authentication 41
3.6.3.4 Linking and Verification 41
3.6.4 Card Management 42
3.6.5 Services 43
4. Security objectives (ASE_OBJ) 45
4.1 Security objectives for the TOE 45
4.1.1 Security Objectives for the TOE not contained in [1] 46
4.1.1.1 Security Objectives for the complete TOE 46
4.1.1.2 Additional Security Objectives for the IC 47
4.1.1.3 Security Objective concerning Random Numbers 48
4.1.2 Security Objectives for the TOE from [1] 48
4.1.2.1 Identification 48
4.1.2.2 Execution 48
4.1.2.3 Services 49
4.1.2.4 Card Management 49
4.1.2.5 Smart Card Platform 50
4.2 Security objectives for the operational environment 50
4.2.1 Security Objectives for the operational environment not contained in [1] 51
4.2.1.1 Objectives on Phase 7 51
4.2.2 Security Objectives for the operational environment from [1] 51
4.3 Relation between security objectives and the security problem definition 52
4.3.1 Justifications from [1] 56
4.3.2 Justification from [4] 58
5. Extended Components Definition (ASE_ECD) 58
5.1 Definition of Family FCS_RNG 58
5.2 Definition of the Family FPT_EMSEC 59
5.3 Definition of the Family FMT_LIM 60
5.4 Definition of Family FAU_SAS 61
6. Security requirements (ASE_REQ) 62
6.1 Users & Subjects 62
6.2 Security functional requirements (SFRs) 63
6.2.1 Firewall Policy 68
6.2.1.1 FDP_ACC.2/FIREWALL Complete Access Control 68
6.2.1.2 FDP_ACF.1/FIREWALL Security Attribute based Access Control 69
6.2.1.3 FDP_IFC.1/JCVM Subset Information Flow Control 72

continued >>

6.2.1.4 FDP_IFF.1/JCVM Simple Security Attributes ... 73
6.2.1.5 FDP_RIP.1 Subset Residual Information Protection ... 74
6.2.1.6 FMT_MSA.1/JCRE Management of Security Attributes ... 74
6.2.1.7 FMT_MSA.2/JCRE Secure Security Attributes 74
6.2.1.8 FMT_MSA.3/FIREWALL Static Attribute Initialization ... 75
6.2.1.9 FMT_SMR.1/JCRE Security roles ... 75
6.2.1.10 TSF domain separation ... 75
6.2.2 Application Programming Interface ... 75
6.2.2.1 FCS_CKM.1 Cryptographic KEY Generation ... 76
6.2.2.2 FCS_CKM.2 Cryptographic KEY Distribution ... 76
6.2.2.3 FCS_CKM.3 Cryptographic KEY Access ... 76
6.2.2.4 FCS_CKM.4 Cryptographic KEY Destruction ... 76
6.2.2.5 FCS_COP.1 Cryptographic Operation ... 76
6.2.2.6 FDP_RIP.1 Subset Residual Information Protection ... 78
6.2.2.7 FDP_ROL.1/FIREWALL Basic Rollback ... 79
6.2.3 Card Security Management ... 79
6.2.3.1 FAU_ARP.1/JCS Security Alarms ... 80
6.2.3.2 FDP_SDI.2 Stored Data Integrity Monitoring and Action ... 81
6.2.3.3 FPT_RVM.1 Reference Mediation ... 81
6.2.3.4 FPT_FLS.1/JCS Failure with Preservation of Secure State ... 81
6.2.3.5 FPR_UNO.1 Unobservability ... 81
6.2.3.6 FPT_TST.1 TSF Testing ... 82
6.2.4 AID Management ... 82
6.2.4.1 FMT_MTD.1/JCRE Management of TSF Data. 82
6.2.4.2 FMT_MTD.3 Secure TSF data ... 82
6.2.4.3 FIA_ATD.1/AID User Attribute Definition ... 82
6.2.4.4 FIA_UID.2/AID User Identification before any Action ... 83
6.2.4.5 FIA_USB.1 User-Subject binding ... 83
6.2.5 SCPG Security Functional Requirements ... 83
6.2.5.1 FPT_AMT.1 Abstract Machine Testing ... 83
6.2.5.2 FPT_FLS.1 Failure with preservation of a Secure State ... 84
6.2.5.3 FRU_FLT.2/SCP Limited Fault Tolerance ... 84
6.2.5.4 FPT_PHP.3/SCP Resistance to Physical Attack ... 84
6.2.5.5 FDP_ACC.1/SCP Subset access control ... 85
6.2.5.6 FDP_ACF.1/SCP Security attribute based access control ... 85
6.2.5.7 FDP_RIP.1/SCP Subset information flow control ... 86
6.2.5.8 FDP_IFC.1 Subset information flow control ... 87
6.2.5.9 FDP_ITT.1/SCP Basic internal transfer protection ... 87
6.2.5.10 FPT_ITT.1/SCP Basic internal TSF data transfer protection ... 87
6.2.5.11 FPT_SEP.1 TSF Domain Separation ... 87
6.2.5.12 FPT_RCV/SCP Trusted Recovery ... 87
6.2.5.13 FPT_RVM.1 Reference Mediation ... 88
6.2.5.14 FMT_MSA.3/SCP Static attribute initialisation ... 88
6.2.6 CMGRG Security Functional Requirements ... 89
6.2.6.1 FDP_ACC.1/CMGR Subset Access Control ... 89
6.2.6.2 FDP_ACF.1/CMGR Security Attribute based Access Control ... 89
6.2.6.3 FMT_MSA.1/CMGR Management of Security Attributes ... 90
6.2.6.4 FMT_MSA.3/CMGR Static Attribute Initialization ... 90
6.2.6.5 FMT_SMR.1/CMGR Security Roles ... 90
6.2.6.6 FIA_UID.1/CMGR Timing of Identification ... 90
6.2.7 Further Functional Requirements not contained in [1] ... 90
6.2.7.1 FDP_ETC.1 Export of User Data without Security Attributes ... 90
6.2.7.2 FDP_ITC.1 Import of User Data without Security Attributes ... 90
6.2.7.3 FIA_AFL.1 Basic authentication Failure Handling ... 91
6.2.7.4 FIA_UAU.1 Timing of Authentication ... 91
6.2.7.5 FIA_UAU.3/CMGR Unforgeable Authentication ... 92
6.2.7.6 FIA_UAU.4/CMGR Single-use Authentication Mechanisms ... 92
6.2.7.7 FTP_ITC.1/CMGR Inter-TSF Trusted Channel – none ... 92
6.2.7.8 FAU_SAA.1 Potential Violation Analysis ... 92
6.2.7.9 FAU_SAS.1/SCP Audit Data Storage ... 93
6.2.7.10 FMT_SMF.1 Specification of Management Function ... 93
6.2.7.11 FCS_RNG.1 Quality metric for Random Numbers ... 93
6.2.7.12 FPT_EMSEC.1 TOE Emanation ... 94
6.2.7.13 FMT_LIM.1 Limited Capabilities ... 94
6.2.7.14 FMT_LIM.2 Limited Availability ... 94
6.2.7.15 FPT_PHP.1 Passive Detection of physical Attack ... 94
6.2.7.16 FPT_TDC.1 Inter-TSF basic TSF Data Consistency ... 94
6.3 Security assurance requirements (SARs) ... 95
6.4 SARs and the security requirement rationale ... 97
6.4.1 Security Functional Requirements Rationale ... 97
6.4.1.1 TOE Security Requirements Rationale ... 97
6.4.2 SARs rationale ... 101
6.4.2.1 Evaluation Assurance Level Rationale ... 101

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

Date of release: 08 December 2010

6.4.2.2 Assurance Augmentations Rationale 101
7. TOE summary specification (ASE_TSS) 101
7.1 Security Functionality 101
7.1.1 SF.AccessControl 102
7.1.2 SF.Audit 103
7.1.3 SF.CryptoKey 105
7.1.4 SF.CryptoOperation 106
7.1.5 SF.I&A 107
7.1.6 SF.SecureManagment 107
7.1.7 SF.PIN 108
7.1.8 SF.Transaction 109
7.1.9 SF.Hardware 109
7.1.10 SF.CryptoLib 109
8. Bibliography 111
9. Legal information 113
9.1 Definitions 113
9.2 Disclaimers 113
10. Contents 114

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

Date of release: 08 December 2010