STMicroelectronics

J-Safe on SB23YR80B Security Target - Public Version

Common Criteria for IT security evaluation

J-SAFE_on_SB23YR80B_ST_Lite_A
April 2015

BLANK

J-Safe on SB23YR80B
Security Target - Public Version

Common Criteria for IT Evaluation

1. Introduction

1.1 Document Reference

Document identification: **J-Safe on SB23YR80B Security Target - Public Version**
Revision: **A**
Registration: **J-SAFE_on_SB23YR80B_ST_Lite_A**

1.2 Security Target Reference

Document identification: J-Safe on SB23YR80B Security Target
Revision: G
Registration: J-Safe_SB23YR80B_SecurityTarget_G
Doc. ID: 8383811

1.3 TOE Reference

Product Name: **J-Safe ID 80K**
TOE Name and Version: **J-Safe on SB23YR80B v.2.11.0**
TOE Hardware: **SB23YR80B (Maskset K2M0A, external Rev. B, internal Rev. H or I)** (ref. ANSSI-CC-2012/68).

1.4 Purpose

This document details the **Security Target – Public Version** of **J-Safe Java Card Platform** designed on the **ST23 SB23YR80B platform (ST23YR80 Security Integrated Circuit with dedicated software and embedded cryptographic library)**.

This document is a sanitized version of the Security Target ([J-SAFE_ST]) used for the evaluation. It is classified as public information.

The precise description of the Target of Evaluation (TOE) and the related features are given in chapter 3 - TOE Description.

A glossary of terms and abbreviations used in this document is given in chapter 3 - DEFINITIONS.

INDEX

List of tables

List of figures

2. REFERENCE DOCUMENTS

2.1 Other reference documents

[CC1]	Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1. Revision 3. July 2009. CCMB-2009-07-001.
[CC2]	Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1. Revision 3. July 2009. CCMB-2009-07-002.
[CC3]	Common Criteria for Information Technology Security Evaluation, Part 3:Security assurance requirements. Version 3.1. Revision 3. July 2009. CCMB-2009-07-003.
[CEM]	Common Methodology for Information Technology Security Evaluation, Evaluation Methodology. Version 3.1. Revision 3. July 2009. CEM-2009-07-004.
[CSRS]	GlobalPlatform Card Security Requirements Specification, Version 1.0,May 2003.
[GPSCSTG]	GlobalPlatform Smart Card Security Target Guidelines, V1.0 October 2005.
[JCVM22]	Java Card Platform, version 2.2 Virtual Machine (Java Card VM) Specification. June 2002. Published by Sun Microsystems, Inc.
[JCAPI22]	Java Card Platform, version 2.2 Application Programming Interface. June 2002. Published by Sun Microsystems, Inc.
[JCRE22]	Java Card Platform, version 2.2 Runtime Environment (Java Card RE) Specification. June 2002. Published by Sun Microsystems, Inc.
[JCAPI222]	Application Programming Interface, Java Card™ Platform, Version 2.2.2, March 2006, Sun Microsystems, Inc.
[JCRE222]	Runtime Environment Specification, Java Card™ Platform, Version 2.2.2, March 2006, Sun Microsystems, Inc.
[JCVM222]	Virtual Machine Specification, Java Card™ Platform, Version 2.2.2, March 2006, Sun Microsystems, Inc.
[JCVM3]	Java Card Platform, versions 3.0 (March 2008), 3.0.1 (April 2009) and 3.0.4 (September 2011), Classic Edition, Virtual Machine (Java Card VM) Specification. Published by Sun Microsystems, Inc.
[JCAPI3]	Java Card Platform, versions 3.0 (March 2008), 3.0.1 (April 2009) and 3.0.4 (September 2011), Classic Edition, Application Programming Interface, March 2008. Published by Sun Microsystems, Inc.
[JCRE3]	Java Card Platform, versions 3.0 (March 2008), 3.0.1 (April 2009) and 3.0.4 (September 2011), Classic Edition, Runtime Environment (Java Card RE) Specification. March 2008. Published by Sun Microsystems, Inc.
[JCBV]	Java Card Platform, version 2.2 Off-Card Verifier. June 2002. Whitepaper. Published by Sun Microsystems, Inc.
[JAVASPEC]	The Java Language Specification. Third Edition, May 2005. Gosling, Joy, Steele and Bracha. ISBN 0-321-24678-0.
[JVM]	The Java Virtual Machine Specification. Lindholm, Yellin. ISBN 0-201-43294-3.
[GP211]	GlobalPlatform Card Specification, Version 2.1.1, March 2003.

[VGP]	Visa GlobalPlatform Card Implementaion Requirements, March 2007.
[PP-JCS-1.0]	Java Card Protection Profile Collection, Version 1.0b, August 2003, registered and certified by the French certification body (ANSSI) under the following references: [PP/0303] “Minimal Configuration”, [PP/0304] “Standard 2.1.1 Configuration”, [PP/0305] “Standard 2.2 Configuration” and [PP/0306] “Defensive Configuration”.
[PP_JC_Closed]	Java Card Protection Profile – Closed Configuration, Version 3.0, December 2012
[PP_ESforSSD]	Embedded Software for Smart Secure Devices Protection Profile, v1.0, November 27th 2009, ANSSI.
[PP_0035]	Security IC Platform Protection Profile, Version 1.0, 15 July 2007.
[STlite_SB23]	SA23YR48B / SB23YR48B / SA23YR80B / SB23YR80B SECURITY TARGET - PUBLIC VERSION, Rev 03.00, March 2011
[BSI_AIS31]	A proposal for Functionality classes and evaluation methodology for true (physical) random number generators, W. Killmann & W. Schindler. BSI, Version 3.1, 25-09-2001
[BSI_AIS20]	Functionality classes and evaluation methodology for deterministic random number generators, BSI, Version 1, 02-12-1999
[PP_ICAO_BAC]	Protection Profile Machine Readable travel Document with “ICAO Application”, Basic Access Control, Version 1.10, 25th March 2009 (BSI-CC-PP-0055)
[FIPS_197]	FIPS Publication 197, ADVANCED ENCRYPTION STANDARD (AES), U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, November 26, 2001
[FIPS_46-3]	FIPS Publication 46-3, DATA ENCRYPTION STANDARD (DES), Reaffirmed 1999 October 25, U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology
[ANSI_X9.62]	ANSI X9.62-2005: The Elliptic Curve Digital Signature Algorithm (ECDSA), approved November 16, 2005
[RSA-PKCS1]	PKCS #1: RSA Encryption Standard - An RSA Laboratories Technical Note, Version 1.5, Revised November 1, 1993
[FIPS_180-2]	FIPS Publication 180-2: SECURE HASH STANDARD, U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, 2002 August 1
[ISO_9797-1]	ISO/IEC 9797-1:1999: Information technology - Security techniques – Message Authentication Codes (MACs) - Part 1: Mechanisms using a block cipher
[ISO_9796-2]	ISO/IEC 9797-2:2002: Information technology - Security techniques – Digital Signature Scheme - Part 2: Integer factorization based mechanisms
[RSA_PSS]	RSA Laboratories. PKCS#1 v2.1: RSA cryptography standard, RSA Laboratories Technical Note, 2002
[PKCS3]	PKCS #3: Diffie-Hellman Key-Agreement Standard Version 1.4 Revised November 1, 1993
[IEEEP1363]	Standard Specifications for Public Key Cryptography, Institute of Electrical and Electronic Engineers, 2000 : IEEE 1363
[J-SAFE_ST]	J-Safe on SB23YR80B Security Target – Rev. G, January 2015, Doc. ID: 8383811

3. DEFINITIONS

Term	Meaning
0x	C-fashion hexadecimal prefix
A.XXX	Assumption
AID	Application identifier, an ISO-7816 data format used for unique identification of Java Card applets (and certain kinds of files in card file systems). The Java Card platform uses the AID data format to identify applets and packages. AIDs are administered by the International Opens Organization (ISO), so they can be used as unique identifiers. AIDs are also used in the security policies (see “Context” below): applets’ AIDs are related to the selection mechanisms, packages’ AIDs are used in the enforcement of the firewall. Note: although they serve different purposes, they share the same namespace.
APDU	Application Protocol Data Unit, an ISO 7816-4 defined communication format between the card and the off-card applications. Cards receive requests for service from the CAD in the form of APDUs. These are encapsulated in Java Card System by the javacard.framework.APDU class ([JCAPI22]). APDUs manage both the selection-cycle of the applets (through Java Card RE mediation) and the communication with the Currently selected applet.
APDU buffer	The APDU buffer is the buffer where the messages sent (received) by the card depart from (arrive to). The Java Card RE owns an APDU object (which is a Java Card RE Entry Point and an instance if the javacard.framework.APDU class) that encapsulates APDU messages in an internal byte array, called the APDU buffer. This object is made accessible to the currently selected applet when needed, but any permanent access (out-of selection-scope) is strictly prohibited for security reasons.
Applet	The name given to any Java Card technology-based application. An applet is the basic piece of code that can be selected for execution from outside the card. Each applet on the card is uniquely identified by its AID.
Applet deletion ager	The on-card component that embodies the mechanisms necessary to delete an applet or library and its associated data on smart cards using Java Card technology.
BCV	The bytecode verifier is the software component performing a static analysis of the code to be loaded on the card. It checks several kinds of properties, like the correct format of CAP files and the enforcement of the typing rules associated to bytecodes. If the component is placed outside the card, in a secure environment, then it is called an off-card verifier. If the component is part of the embedded software of the card it is called an on-card verifier.
CAD	Card Acceptance Device or card reader. The device where the card is inserted, and which is used to communicate with the card. Unless explicitly said otherwise, in this document, CAD covers PCD.
CAP file	A file in the Converted applet format. A CAP file contains a binary representation of a package of classes that can be installed on a device and used to execute the package’s classes on a Java Card virtual machine. A CAP file can contain a user library, or the code of one or more applets.
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 Manager
Context	A context is an object-space partition associated to a package. Applets within the same Java technology-based package belong to the same context. The firewall is the boundary between contexts (see “Current context”).
Current Context	The Java Card RE keeps track of the current Java Card System context (also called “the active context”). When a virtual method is invoked on an object, and a context switch is required and permitted, the current context is changed to correspond to the context of the applet that owns the object. When that method returns, the previous context is restored. Invocations of static methods have no effect on the current context. The current context and sharing status of an object together determine if access to an object is permissible.
Currently Selected Applet	The applet has been selected for execution in the current session. The Java Card RE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command from the CAD or PCD with this applet’s AID, the Java Card RE makes this applet the currently selected applet over the I/O interface that received the command. The Java Card RE sends all further APDU commands received over each interface to the currently selected applet on this interface ([JCRE3], Glossary).
Default Applet	The applet that is selected after a card reset or upon completion of the PICC activation sequence on the contactless interface ([JCRE3], §4.1)
DPA	Differential Power Analysis is a form of side channel attack in which an attacker studies the power consumption of a cryptographic hardware device such as a smart card.
EAL	Evaluation Assurance Level
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 Java Card RE or by an applet in another context.
HAL	Hardware Abstraction Layer
IC	Integrated Circuit
Installer	The installer is the on-card application responsible for the installation of applets on the card. It may perform (or delegate) mandatory security checks according to the card issuer policy (for bytecode-verification, for instance), loads and link packages (CAP file(s)) on the card to a suitable form for the Java Card VM to execute the code they contain. It is a subsystem of what is usually called “card manager”; as such, it can be seen as the portion of the card manager that belongs to the TOE. The installer has an AID that uniquely identifies him, and may be implemented as a Java Card applet. However, it is granted specific privileges on an implementation-specific manner ([JCRE3],§10). The installer is the on-card application responsible for the installation of applets on the card. It may perform (or delegate) mandatory security checks according to the card issuer policy (for bytecode-verification, for instance), loads and link packages (CAP file(s)) on the card to a suitable form for the Java Card VM to execute the code they contain. It is a subsystem of what is usually called “card manager”; as such, it can be seen as the portion of the card manager that belongs to the TOE. The installer has an AID that uniquely identifies him, and may be implemented as a Java Card applet. However, it is granted

	specific privileges on an implementation-specific manner ([JCRE3],§10).
Interface	A special kind of Java programming language class, which declares methods, but provides no implementation for them. A class may be declared as being the implementation of an interface, and in this case must contain an implementation for each of the methods declared by the interface (See also shareable interface).
Java Card RE	The runtime environment under which Java programs in a smart card are executed. It is in charge of all the management features such as applet lifetime, applet isolation, object sharing, applet loading, applet initializing, transient objects, the transaction mechanism and so on.
Java Card RE Entry Point	An object owned by the Java Card RE context but accessible by any application. These methods are the gateways through which applets request privileged Java Card RE services: the instance methods associated to those objects may be invoked from any context, and when that occurs, a context switch to the Java Card RE context is performed. There are two categories of Java Card RE Entry Point Objects: Temporary ones and Permanent ones. As part of the firewall functionality, the Java Card RE detects and restricts attempts to store references to these objects.
Java Card RMI	Java Card Remote Method Invocation is the Java Card System version 2.2 and 3 Classic Edition mechanism enabling a client application running on the CAD platform to invoke a method on a remote object on the card. Notice that in Java Card System, version 2.1.1, the only method that may be invoked from the CAD is the process method of the applet class.
Java Card System	Java Card System includes the Java Card RE, the Java Card VM, the Java Card API and the installer.
Java Card VM	The embedded interpreter of bytecodes. The Java Card VM is the component that enforces separation between applications (firewall) and enables secure data sharing.
Logical Channel	A logical link to an application on the card. A new feature of the Java Card System, version 2.2 and 3 Classic Edition, that enables the opening of simultaneous sessions with the card, one per logical channel. Commands issued to a specific logical channel are forwarded to the active applet on that logical channel. Java Card platform, version 2.2.2 and 3 Classic Edition, enables opening up to twenty logical channels over each I/O interface (contacted or contactless).
NVM	Non-Volatile Memory (see NVRAM)
NVRAM	Non-Volatile Random Access Memory, a type of memory that retains its contents when power is turned off.
O.xxx	Security objectives for the TOE.
Object Deletion	The Java Card System version 2.2 and 3 Classic Edition mechanism ensures that any unreferenced persistent (transient) object owned by the current context is deleted. The associated memory space is recovered for reuse prior to the next card reset.
OE.xxx	Security objectives for the environment.
OSP.xxx	Organizational security policies.
Package	A package is a namespace within the Java programming language that may contain classes and interfaces. A package defines either a user library, or one or more applet definitions. A package is divided in two sets of files: export files (which exclusively contain the public interface information for an entire package of classes, for external linking purposes; export files are not used directly in a Java Card virtual machine) and CAP files.
PCD	Proximity Coupling Device. The PCD is a contactless card reader device.
PICC	Proximity Card. The PICC is a card with contactless capabilities.

PP	Protection Profile.
RAM	Random Access Memory, is a type of computer memory that can be accessed randomly.
ROM	Read Only Memory.
SC	Smart Card
SCP	Smart Card Platform.It is comprised of the integrated circuit, the operating system and the dedicated software of the smart card.
SF.xxx	Security Functionality
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.
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.
SWP	The Single Wire Protocol is a specification for a single-wire connection between the SIM card and a Near Field Communication (NFC) chip in a mobile handset
TOE	Target Of Evaluation
Transient Object	An object whose contents are not preserved across CAD sessions. The contents of these objects are cleared at the end of the current CAD session or when a card reset is performed. Writes to the fields of a transient object are not affected by transactions.
User	Any application interpretable by the Java Card RE. That also covers the packages. The associated subject(s), if applicable, is (are) an object(s) belonging to the javacard.framework.applet class.
VM	Virtual Machine

4. TOE Description

4.1.1 TOE overview

4.1.1.1 TOE TYPE

J-Safe ID 80K is a EAL5+ certified smart card product targeting the Financial and ID market segments and defined to be used as a contact, contactless or dual-interface card, allowing pre-issuance application downloading.

The Target of Evaluation detailed in this Security Target is **J-Safe on SB23YR80B v.2.11.0** (from now on also referenced as **J-Safe v.2.11.0** or, simply, the TOE): a Java Card-enabled smart card platform embedded in **J-Safe ID 80K** product.

The TOE is constituted by the following blocks:

- The Java Card System according to [PP_JC_Closed], which manages and executes application called applets. It also provides API ([JCAPI3]) to develop applets on top of it in accordance with Java Card^{TM} specifications;
- The GlobalPlatform API ([GP211]);
- Additional proprietary API to support ID-specific requirements;
- A native Operating system providing to the Java Card System a low-level support of hardware functionalities and implementing I/O communication;
- The hardware IC and its associated crypto library (already certified as SB23YR80B).

The TOE fulfils all the requirements of [PP_JC_Closed] and adds some proprietary extensions due to product-specific needs. Extension will be detailed later on in the document.

This Security Target selects a hierarchically higher assurance level (EAL) with respect to the one defined by the PP:

- Assurance Level: **EAL5+** (instead of EAL4+)
- Augmentations: **ALC_DVS.2 and AVA_VAN.5**.

J-Safe v.2.11.0 is based on Java Card 3.0.4 Classic Edition ([JCRE3], [JCAPI3], [JCVM3]) and GlobalPlatform 2.1.1 ([GP211]) industry standards. Moreover it is fully compliant with “Configuration 2 – Compact with Public Key” requirements of VISA GlobalPlatform Card Implementation Requirements ([VGP]).

J-Safe v.2.11.0 provides the following main features:

- Communication protocols:
  - T=0
  - T=1
  - T=CL (contact-less)

- Cryptographic algorithms and services:
  - DES / 3-DES
  - AES (up to 256 bits)
  - RSA Standard and CRT with key generation (up to 2048 bits)
  - SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
  - EC over GF(p) in the range between 160 and 521 bits
  - Secure random number generation

- Card and Card Content Administration during pre-issuance phase:
  - User authentication and Secure Communication using Secure Channel Protocol 02 (SCP02);
  - Card Life Cycle management;
  - Downloading, installation and deletion of applications (Java Card applets) operations.

4.1.2 TOE description

4.1.2.1 TOE Boundaries

The Target of Evaluation (TOE) is the Java Card-enabled smart card platform embedded in **J-Safe ID 80K** product.

Physical Boundaries

The TOE is constituted by hardware and software parts and is available in several formats depending on the product end usage:

- Contact-only card (IC packaged as micro-module and embedded in a plastic card body)
- Dual-interface card (IC packaged as micro-module and antenna embedded in a plastic card body)
- Contactless-only card (IC chip integrated with antenna on an inlay and embedded in a plastic card body)
- DFN-8 modules (IC packaged in DFN-8 format for integration on PCBs)
- Wafers or sawn wafers (e.g.: to be embedded on paper sheets - contactless inlays)

The carrier, being it made in paper or plastic, and the optional antenna are out of the scope of current evaluation.

Logical Boundaries

The Target of Evaluation (TOE) is the Java Card-enabled smart card platform embedded in **J-Safe ID 80K** product according to Figure 1.

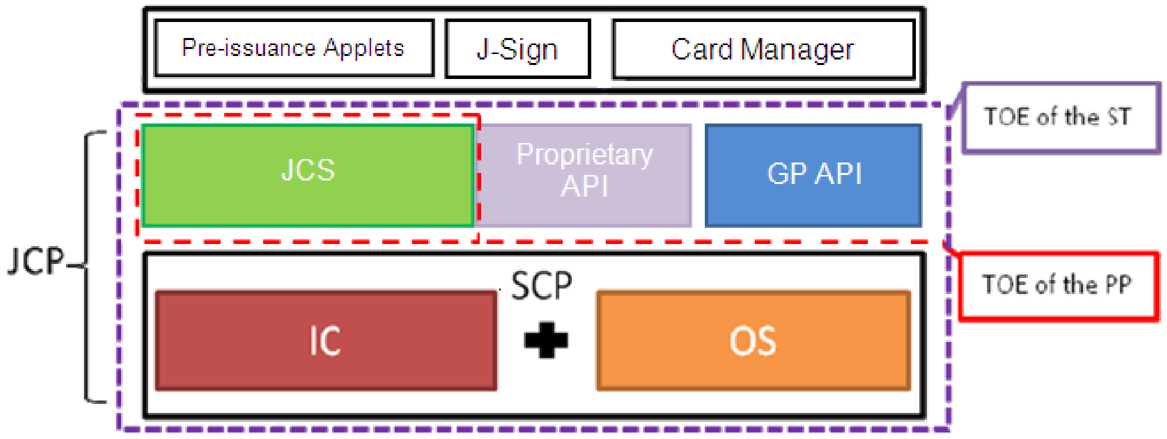


Figure 1 - **J-Safe ID 80K** product with TOE and JCS PP boundaries

According to the terminology of [PP_JC_Closed], the Secure IC hardware (from now on simply called IC), its associated crypto-library, and the native operating system are ideally combined in the so called Smart Card Platform (SCP).

The **J-Safe ID 80K** product includes the GlobalPlatform Card Manager applet, which provides, according to [GP211] specifications and [VGP] requirements, a common and widely used administrative interface for the secure management of applications on the smart card. The Card Manager is out of scope of current evaluation and its administrative interface is permanently disabled before TOE delivery.

The **J-Safe ID 80K** product also stores the “J-Sign” ID application: such application is a Java Card applet embedded in ROM and is out of the scope of current evaluation.
Any other application being loaded pre-issuance is out of scope of current evaluation.

4.1.2.1 Java Card Platform

The Java technology, embedded on the TOE, combines a subset of the Java programming language with a runtime environment optimized for smart cards and similar small-memory embedded devices [JCVM3]. The Java Card platform is a smart card platform enabled with Java Card technology (also called, for short, a “Java Card”). This technology allows for multiple applications to run on a single card and provides facilities for secure interoperability of applications. Applications running on the Java Card platform ("Java Card applications") are called applets.

The Java Card VM is the bytecode interpreter as specified in [JCRE3], [JCAPI3], [JCVM3].
The Java Card RE is responsible for card resource management, communication, applet execution, on-card system and applet security. The Java Card API provides classes and interfaces to the Java Card applets. It defines the calling conventions by which an applet may access the Java Card RE and native services such as, I/O management functions, PIN and cryptographic specific management and the exceptions mechanism.

While the Java Card VM is responsible for ensuring language-level security, the Java Card RE provides additional security features for Java Card technology-enabled devices. Applets from different vendors can coexist in a single card, and they can even share information. In the Java Card platform, applet isolation is achieved through the applet firewall mechanism (JCRE3 §6.1). That mechanism confines an applet to its own designated memory area, thus each applet is prevented from accessing fields and operations of objects owned by other applets, unless a "shareable interface" is explicitly provided (by the applet who owns it) for allowing access to that information.

The Java Card RE allows sharing using the concept of "shareable interface objects" (SIO) and static public variables. Java Card VM dynamically enforces the firewall, that is, at runtime. However applet isolation cannot be entirely granted by the firewall mechanism if certain integrity conditions are not satisfied by the applications loaded on the card. Those conditions can be statically verified to hold by a bytecode verifier (JCRE3).

The Java Card VM ensures that the only way for applets to access any resources are either through the Java Card RE or through the Java Card API (or other vendor-specific APIs). This objective can only be guaranteed if applets are correctly typed (all the "must clauses" imposed in chapter 7 of JCRE3 on the bytecodes and the correctness of the CAP file format are satisfied).

On current TOE the Java Card Platform also includes a set of proprietary API providing optimized services for handling integrity of application-specific sensitive data.

4.1.2.1 GlobalPlatform

The **J-Safe ID 80K** product is compliant with the GlobalPlatform 2.1.1 (GP) standard [GP211] which provides a set of APIs and technologies to perform, in a secure way, the operations involved in the management of the applications hosted by the card.
Using GP maximizes the compatibility and the opportunities of communication as it has become the current card management standard.

The main features addressed by GP are:

- card content management operations (i.e. the downloading, installation, removal) of Java Card applications;
- authentication of users through secure channels;
- life-cycle management of both the card and the applications;
- sharing of a global common PIN among all the applications installed on the card.

Card Content Management operations are only available through a privileged application (the Card Manager) after a successful authentication and are permanently disabled before TOE delivery. Other features are addressed by a set of APIs that can be used by the applications hosted on the card.

The implementation of API and Card Manager available in the **J-Safe ID 80K** product is also compliant with Visa GlobalPlatform Card Implementation Requirements v.2.1.1 [VGP].

NOTE: <u>Being **J-Safe ID 80K** a closed product, only the API subset related to life-cycle management of card and applications are considered as part of the TOE, while all other GP API and the Card Manager application belongs to the TOE environment and are not in the scope of current evaluation.</u>

4.1.2.2 Operating System

The Operating System provides memory management functions, I/O functions that are compliant with ISO standards, transaction facilities and secure (shielded, native) implementation of cryptographic functions.

4.1.2.3 Hardware IC and dedicated crypto library

The basis of this composite evaluation is the STMicroelectronics' SB23YR80B certified product which is, on his turn, a composite evaluation of the ST23YR80 Secured Microcontroller plus the NesLib v.3.1 – Configuration SB crypto-library.

The SB23YR80B Secured Microcontroller with Cryptographic Library has been certified by ANSSI (cert. report ANSSI-CC-2012/68) with assurance level EAL6+: its associated Security Target Lite is [STlite_SB23].

NOTE: Even though the TOE includes the IC and the crypto-library, not all the functionalities of the IC and crypto-library are used and, as a consequence, such unused portions of functionalities are not part of the TOE for current evaluation.

4.1.2.4 TOE Functionalities

The **J-Safe v.2.11.0** provides the following functionalities:

- Communication protocols:
  - ISO 7816 T=0 (direct and inverse convention)
  - ISO 7816 T=1 (direct and inverse convention)
  - ISO 14443 T=CL (contact-less)
  - Extended Length APDUs
- Cryptographic functionalities:
  - 3-DES (112 and 168 bit keys) for encryption/decryption in ECB and CBC mode, MAC generation and verification (CBC-MAC, Retail-MAC)
  - AES (key length 128, 192, 256) for encryption/decryption in ECB and CBC mode
  - RSA (with keys up to 2048 bits) for encryption/decryption, signature verification, key generation in both Standard and CRT mode
  - Message Digest with SHA-1, SHA-224, SHA-256, SHA-512 algorithms
  - Elliptic Curve cryptography over GF(p) for key length between 160 and 521 bits
  - Diffie-Hellman and EC Diffie-Hellman key agreement algorithms
  - Secure random number generation compliant to class with P2 Class of BSI_AIS31
- JC functionalities compliant with [PP_JC_Closed]:
  - Logical Channel awareness (only Basic Logical Channel is supported)
  - Object Deletion (garbage collection) with memory reclamation
  - Application loading, linking and installation operations limited to pre-issuance phase in a controlled environment
- Proprietary functionalities:
  - Secure Storage API (integrity-protected arrays)
  - Secure comparison of byte arrays
  - Generation of random primes

4.1.3 TOE Life-Cycle

The TOE life cycle is part of the product life cycle, i.e. the Java Card platform with applications, which goes from product development to its usage by the final user. The TOE life cycle is fully conformant to the claimed PP. The TOE life cycle phases are those detailed in Figure 2, while the product life cycle phases are detailed in Figure 3. We refer to IC Protection Profile [PP_0035] for a thorough description of Phases 1 to 7:

- Phases 1 and 2 compose the product development: Embedded Software (IC Dedicated Software, Native OS, Java Card System, other platform components such as Card Manager, Applets) and IC development.
- Phase 3 and Phase 4 correspond to IC manufacturing and packaging, respectively. Some IC pre-personalization steps may occur in Phase 3.
- Phase 5 concerns the embedding of software components within the IC.
- Phase 6 is dedicated to the product personalization prior final use.
- Phase 7 is the product operational phase.

The TOE life cycle is composed of four stages:
- Development,
- Storage, pre-personalization and testing
- Personalization and testing
- Final usage.

TOE storage is not necessarily a single step in the life cycle since it can be stored in parts. TOE delivery occurs before storage and may take place more than once if the TOE is delivered in parts. The patches are special TOE parts. These stages map to the typical smartcard life cycle phases as shown in Figure 2.

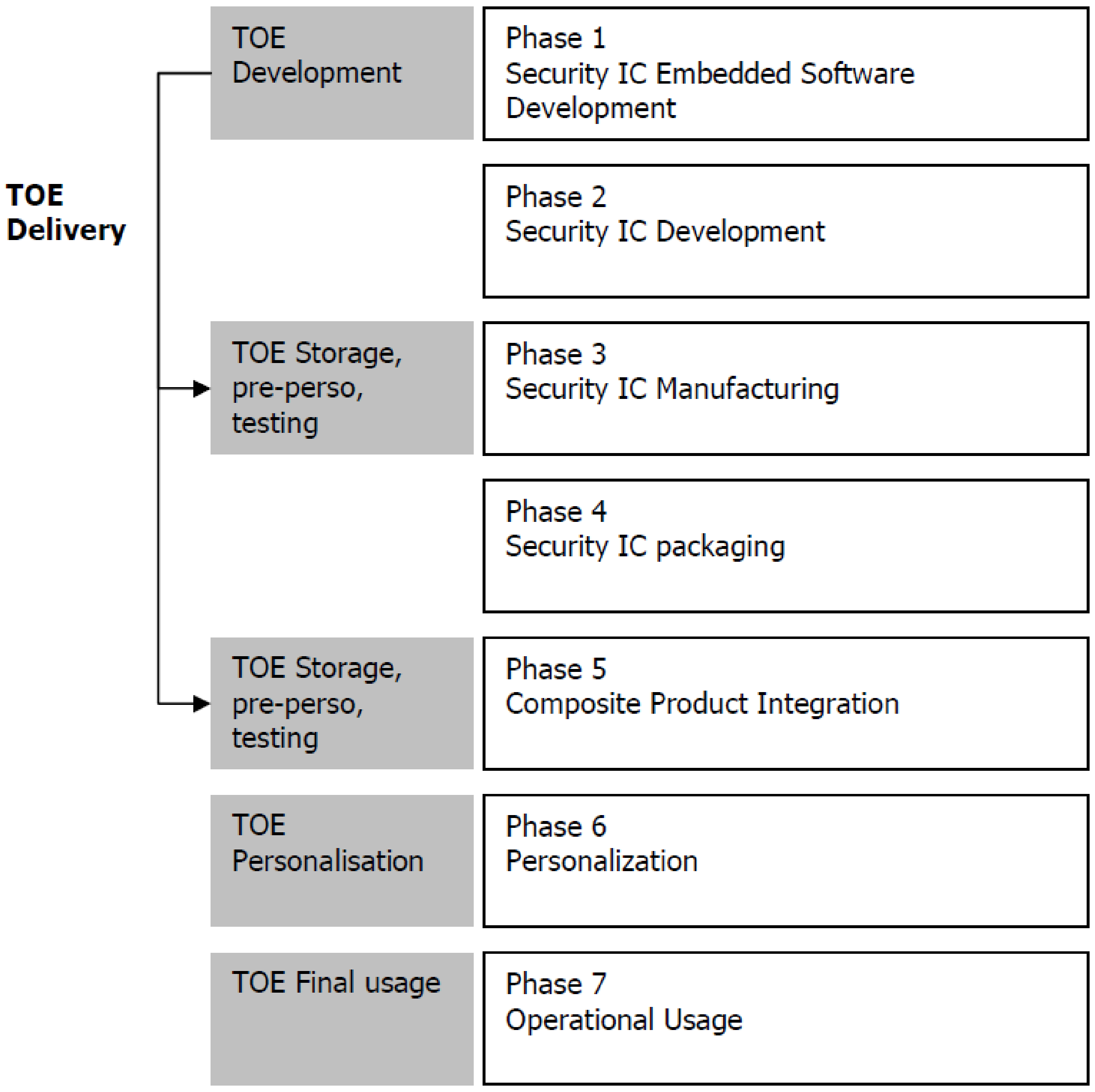

TOE Delivery
TOE Development
Phase 1 Security IC Embedded Software Development
Phase 2 Security IC Development
TOE Storage, pre-perso, testing
Phase 3 Security IC Manufacturing
Phase 4 Security IC packaging
TOE Storage, pre-perso, testing
Phase 5 Composite Product Integration
TOE Personalisation
Phase 6 Personalization
TOE Final usage
Phase 7 Operational Usage

Figure 2 - TOE life cycle

TOE development is performed during Phase 1. This includes Native OS and JCS conception, design, implementation, testing and documentation. The TOE development fulfils requirements of the final product, including conformance to product specifications (e.g. Java Card, GlobalPlatform, etc…), and recommendations of the guidelines of IC and crypto-library. The

TOE development occurs in a controlled environment that avoids disclosure of source code, data and any critical documentation and that guarantees the integrity of these elements. TOE Development is performed by **STMicroelectronics S.r.l – Incard Division** in the site of **MARCIANISE (ITALY)**, from now on referenced as "**STM Incard Division, Marcianise**".

TOE Developer also act as Composite Product Integrator: it shall initialize the TOE by downloading patch code and configuring it according to product needs, then it shall download customer applications on top of the JCS and finally it shall permanently disable card content management features, thus making the TOE fully operational and ready for the delivery.

The delivery of the TOE may occur either during Security IC Manufacturing (Phase 3) or during Composite Product Integration (Phase 5). Delivery and acceptance procedures shall guarantee the authenticity, the confidentiality and integrity of the exchanged pieces. TOE delivery shall involve encrypted signed sending and it supposes the previous exchange of public keys.

In Phase 3, the Security IC Manufacturer may store, pre-personalize the TOE and potentially conduct tests on behalf of the TOE developer. This support by IC Manufacturer is specifically used when the TOE delivery shall occur at the end of phase 3: in this case the TOE Developer may deliver in a secure way to Security IC Manufacturer a NVM image of final product configuration. On its turn, the IC Manufacturer can perform complete TOE pre-personalization on behalf of TOE Developer using the NVM image in order to obtain a fully operational TOE. The Security IC Manufacturing environment shall protect the integrity and confidentiality of the TOE and of any related material, for instance test suites.

In Phase 5, if the expected TOE delivery is at the end of this phase, the TOE Developer shall perform TOE storage, pre-personalization and product configuration steps required as Composite Product Integrator. Being the Composite Product Integrator's environment the same as the TOE Developer's, integrity and confidentiality of the TOE and of any related material are also guaranteed.

According to this life-cycle, at the stage TOE delivery occurs (end of phase 3 or end of phase 5), all the applets of the final product have been installed on top of the JCS (which is part of the TOE), card content management features have been permanently disabled and the TOE has been put in its fully operational state.

In Phase 6, applets of the final product which have been installed on the TOE can be further personalized with end user data, if necessary.

The TOE final usage environment coincides with the environment of the product where the TOE is embedded in. It covers a wide spectrum of situations that cannot be covered by evaluations and, therefore, the TOE and the product shall provide the full set of security functionalities to avoid abuse of the product by un-trusted entities.

Notes on current evaluation:

- Current evaluation process covers phases from 1 to 5; depending on the product format in which the TOE shall be embedded, two possible point of delivery have been defined:
    - end of Phase 3 for TOE delivery in form of wafers/sawn wafers;
    - end of Phase 5 for TOE delivery in form of modules or plastic cards;
- Applet development is not within the scope of this evaluation;
- Depending on the product format, delivery of the TOE can be performed by the following entities:
    - at the end of phase 3 by **qualified ST production sites (see [STlite_SB23]) on behalf of STMicroelectronics S.r.l. - Incard Division**
    - at the end of phase 5 by **STMicroelectronics S.r.l. - Incard Division, MARCIANISE (ITALY)**

- TOE delivery consists in the following items:
    - **J-Safe on SB23YR80B v.2.11.0** (Mask ID: 0x66; ROM Code ID: 0x00020700; Patch Code ID: 0x00021100)
    - Administrator and User Guidance - Rev. D

The delivery is protected by secured transport and tracking measures. TOE identification procedures are described in the guidance documents.

4.1.3.1 Product Life cycle

Here follows the description of the whole product life cycle: it includes all the actual phases and deliveries required during the physical product construction, the pre-issuance administration phase, and the final operational usage (see Figure 3 and Figure 4). Phases 2 and 3 are in charge of the IC Manufacturer and are entirely described in [STlite_SB23]; phase 4 is either in charge of the IC Manufacturer or is performed by STMicroelectronics S.r.l. - Incard Division by subcontracting it to qualified packaging manufacturers.

For complete details on IC development and production centres (generically indicated as STM in Figure 3 and Figure 4), please refer to [STlite_SB23]. Depending on TOE delivery format, IC Packaging may be also performed in **STMicroelectronics – Calamba (PH)** site or by **AMKOR Technologies (PH)** qualified subcontractor.

Typical product life cycle is depicted in Figure 3. This lifecycle applies when customers expect to receive J-Safe TOE volumes in form of modules or plastic cards:

1. IC Manufacturer sends the already packaged ICs to TOE Developer;
2. TOE Developer, within its secure premises, initializes the TOE and, acting as Composite Product Integrator, performs product pre-personalization and, following appropriate validation procedures, it integrates the composite product applets;
3. Once the product configuration is completed (all the applets have been put on top of the JCS), the TOE Developer permanently disables card content management features;
4. TOE Developer also acts as Product Finisher to put the product in its final physical format according to product needs (e.g. plastic card, plastic card with mag-stripe, etc.);
5. TOE Developer performs TOE delivery (end of phase 5).

A variant of the typical life-cycle has been defined to address scenarios where customers expect to receive J-Safe TOE volumes in form of wafers/sawn wafers or when final product format shall be built by external product finishers (see Figure 4). In this case:

1. TOE Developer goes through all necessary steps to obtain the final product configuration (including sensitive operations such as the loading of patches, applets, and permanent disabling of card content management features), generates a product EEPROM image and delivers it to the IC Manufacturer following appropriate delivery procedures;
2. IC Manufacturer, within its premises, performs product initialization and pre-personalization on TOE Developer’s behalf, during “IC pre-personalization and testing” step of phase 3. At the end of this step (end of phase 3), the TOE becomes fully operational and can be safely delivered to package manufacturers, product finishers or customers;

The limits of the evaluation process include the TOE under development until delivery from the party responsible of the construction of the TOE to the parties responsible of the following usage phases. Application development is out of scope of current evaluation.

TOE Construction phase includes:

- Product Development
- Product Manufacturing

TOE usage phase includes:

- Product Personalization
- Product Usage

The different smart card product life cycle phases involve the following different authorities:

TOE Developer	Is responsible for: • Designing the TOE Embedded Software; • Integrating any portion of IC dedicated software required by the TOE delivered by the IC Manufacturer; • Validating and integrating any application (applet) to be masked in ROM together with the TOE Embedded Software; • Initializing the TOE (including download of patch code) according to product needs; • Acts as Composite Product Integrator by creating the final product configuration: - Validating additional applications (e.g. customer

	applets) required for the final product, and, on behalf of the customer, downloading and installing them in EEPROM; - Permanent disabling of Card Content Management services (i.e. Application loading, installation, deletion, and setting of Card Life Cycle); • Optionally, if the chosen TOE Delivery format is wafers/sawn wafers, the TOE Developer delivers the final product configuration in form of NVM image to the IC Manufacturer;
IC Manufacturer	The IC Manufacturer is in charge of designing the IC and developing the IC Dedicated Software (including the crypto library), producing the HW IC. The IC manufacturer may deliver to the TOE Developer some IC Dedicated Software (e.g. the crypto library) that needs to be integrated in the TOE Embedded Software. When the selected format for TOE delivery is “cards”, the IC Manufacturer is also in charge of IC packaging phase. When the selected format for TOE delivery is “wafers/sawn wafers”, the IC Manufacturer, is also in charge of making the TOE fully operational before delivering it to an external Packaging Manufacturer: in this case, the IC Manufacturer performs TOE initialization and full product configuration by using the product image provided by TOE Developer.
Packaging Manufacturer	When the selected format for TOE delivery is “cards”, the IC Manufacturer also acts as Packaging Manufacturer. When the selected format for TOE delivery is “DFN-8 modules”, STMicroelectronics – Calamba site or “AMKOR Technologies” qualified subcontractor is in charge of this phase. Other Packaging Manufacturers act when the TOE (already in its operational state) has been delivered in form of wafers/sawn wafers due to product specific needs. <u>Example</u>: in case of a MRTD product, the TOE is delivered in form of wafers because it shall be embedded on contactless inlays: in this case, the inlay manufacturer (acting as packaging manufacturer) is responsible for the embedding on paper sheets of IC chips received from the IC Manufacturer, performing appropriate physical/electrical testing and for sending them to Product Finisher.
Product Finisher	The Product Finisher may be the TOE Developer itself or a 3rd-party integrator. The Product Finisher is in charge of product finishing: it completes product manufacturing by putting the product into its required physical format according to customer needs. It only acts after the TOE has already been put in its operational state. Examples: - For dual-interface smart cards: integrates inlay and pre-personalized IC module on plastic cards; - For MRTD products: embeds into passport booklets the operational TOE in form of contactless inlays coming from the Inlay Manufacturer.
Personalizer	The Personalizer may be the TOE Developer itself, a 3rd-party Personalizer, or product customer itself and is responsible for:

	• Optionally, personalizing applications and/or the card with the customer' secret data on behalf of the card Issuer (e.g. for a MRTD application, personalization might consist in establishing the identity of the holder with biographical data; the Issuer could be the Issuing State).
End User	The End User is the holder for whom the product has been personalized.

The TOE life cycle is part of the product life cycle, the link between the two life cycles, together with the responsible party in each phase is described in the following table:

Product Phase	TOE Life Cycle Phase	Responsible
Development	Phase 1	TOE Developer
Manufacturing	Phase 2, Phase 3, Phase 4, Phase 5	IC Manufacturer IC Manufacturer IC Manufacturer or Packaging Manufacturer TOE Developer or Product Finisher
Personalization	Phase 6	Personalizer (could be the TOE Developer itself)
Usage	Phase 7	End User

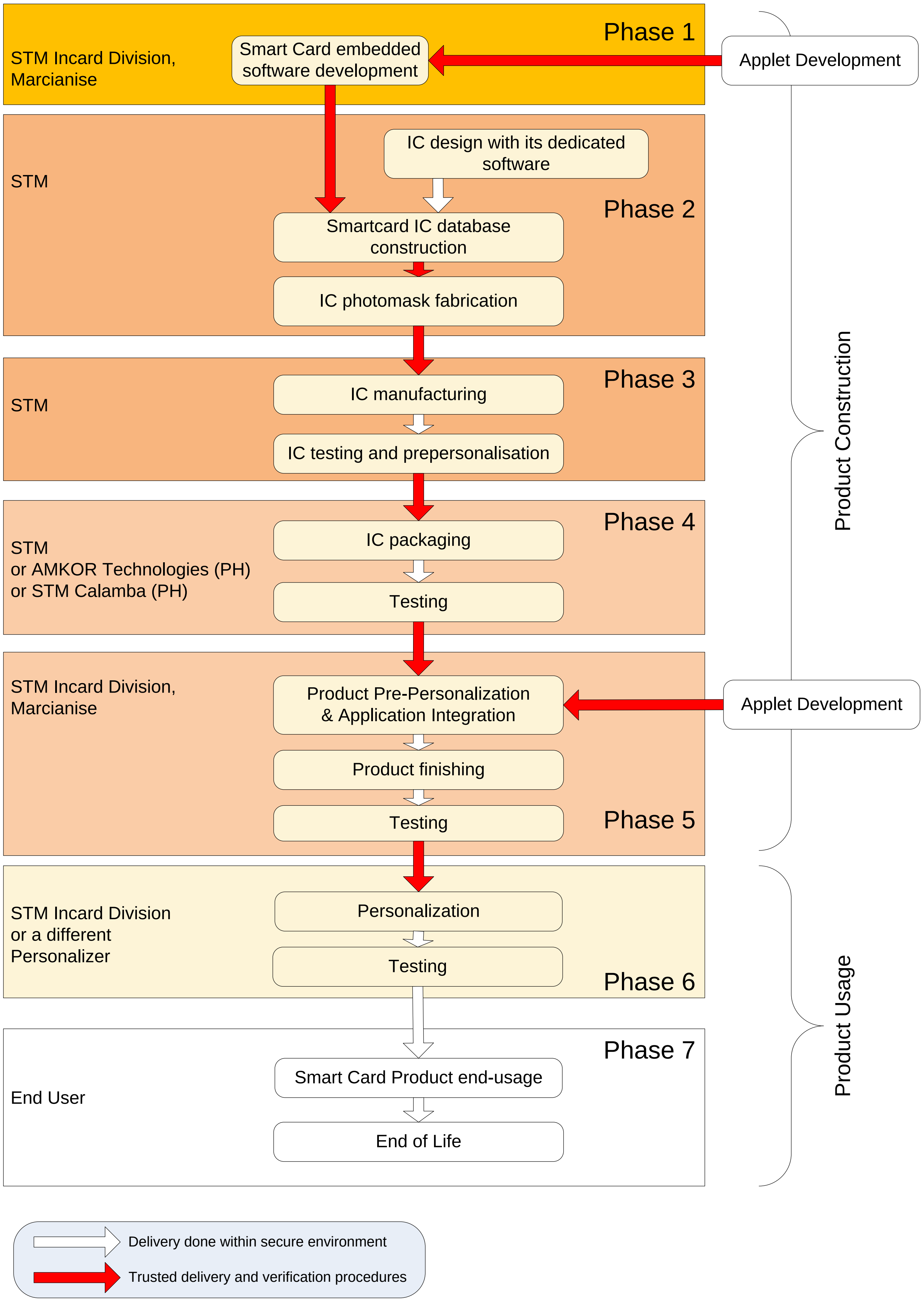

Phase 1
STM Incard Division, Marcianise
Smart Card embedded software development
Applet Development
STM
IC design with its dedicated software
Phase 2
Smartcard IC database construction
IC photomask fabrication
Phase 3
STM
IC manufacturing
IC testing and prepersonalisation
Phase 4
STM or AMKOR Technologies (PH) or STM Calamba (PH)
IC packaging
Testing
STM Incard Division, Marcianise
Product Pre-Personalization & Application Integration
Applet Development
Product finishing
Testing
Phase 5
STM Incard Division or a different Personalizer
Personalization
Testing
Phase 6
Phase 7
End User
Smart Card Product end-usage
End of Life
Product Construction
Product Usage
Delivery done within secure environment
Trusted delivery and verification procedures

Figure 3 – Typical product life cycle with TOE delivery performed at the end of phase 5

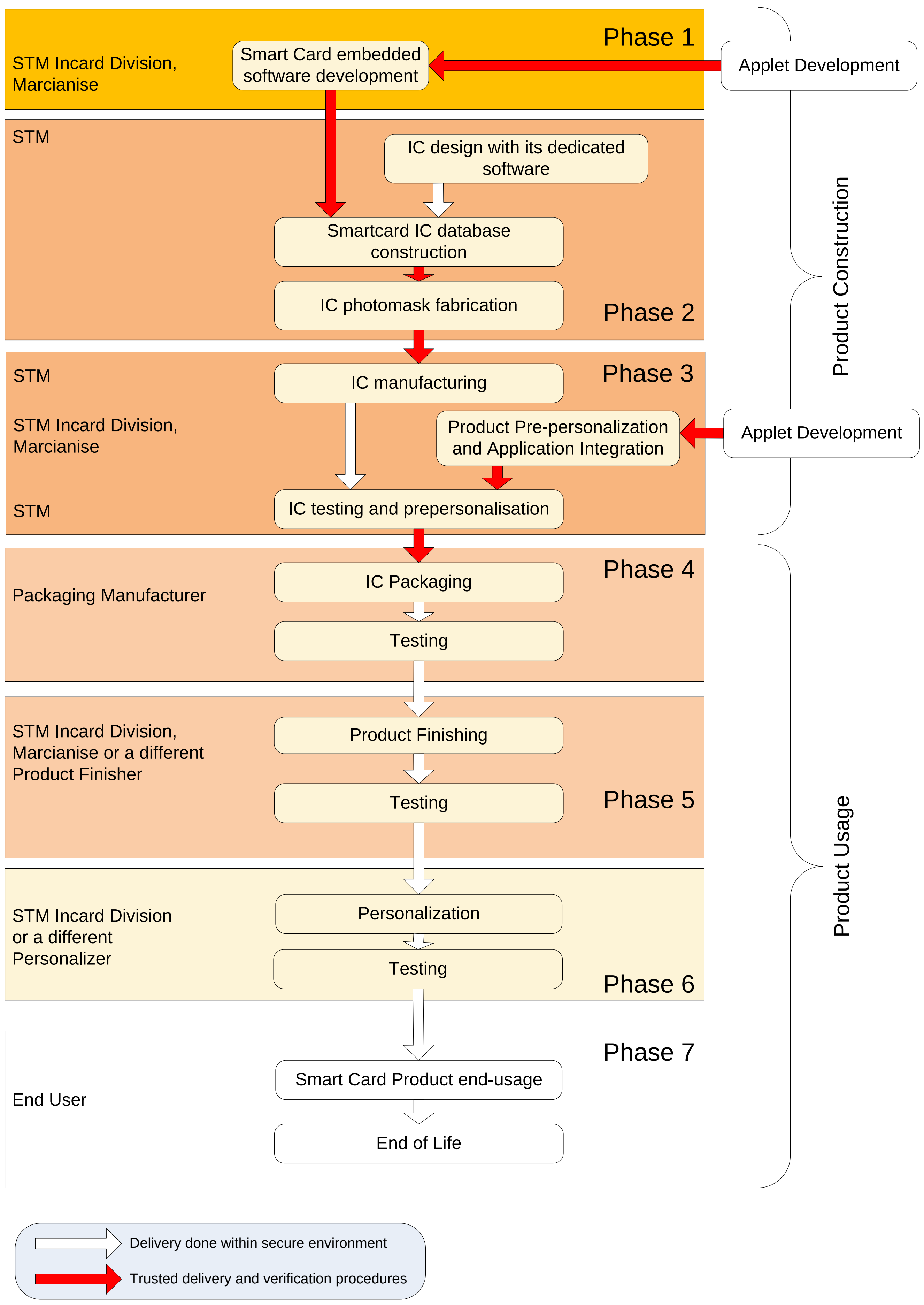

Phase 1
STM Incard Division, Marcianise
Smart Card embedded software development
Applet Development
STM
IC design with its dedicated software
Smartcard IC database construction
IC photomask fabrication
Phase 2
STM
IC manufacturing
Phase 3
STM Incard Division, Marcianise
Product Pre-personalization and Application Integration
Applet Development
STM
IC testing and prepersonalisation
Product Construction
Packaging Manufacturer
IC Packaging
Testing
Phase 4
STM Incard Division, Marcianise or a different Product Finisher
Product Finishing
Testing
Phase 5
STM Incard Division or a different Personalizer
Personalization
Testing
Phase 6
End User
Phase 7
Smart Card Product end-usage
End of Life
Product Usage
Delivery done within secure environment
Trusted delivery and verification procedures

Figure 4 – Product life cycle with TOE delivery performed at the end of phase 3

4.1.4 TOE Intended usage

Smart cards are used as data carriers that are secure against forgery and tampering as well as personal, highly reliable, small size devices capable of replacing paper transactions by electronic data processing. Data processing is performed by a piece of software embedded in the smart card chip, called an application.
The intended use of a Java Card platform is to provide a framework for implementing IC independent applications conceived to safely coexist and interact with other applications into a single smart card.
Applications installed on a Java Card platform can be selected for execution when the card communicates with a card reader.
Notice that these applications may contain other confidentiality (or integrity) sensitive data than usual cryptographic keys and PINs; for instance, passwords or pass-phrases are as confidential as the PIN, or the balance of an electronic purse.
So far, the most typical applications are:

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

4.2 Conformance Claims (ASE_CCL)

4.2.1 CC Conformance Claim

This Security Target and the composite TOE claim conformance to CC version 3.1 to:

- Common Criteria for Information Technology Security Evaluation Version 3.1, Part 2 - "Security functional requirements", extended with FCS_RNG.1, FMT_LIM.1, FMT_LIM.2, FAU_SAS.1, and FPT_EMSEC.1 components drawn from Security IC Platform Protection Profile PP_0035;
- Common Criteria for Information Technology Security Evaluation Version 3.1, Part 3 - Security assurance requirements. The chosen Evaluation Assurance Level is EAL5 augmented with ALC_DVS.2 and AVA_VAN.5;
- Demonstrable conformance with Java Card System – Closed Configuration Protection Profile, Version 3.0, December 2012 ([PP_JC_Closed])

4.2.2 Conformance Claim Rationale

The TOE type consistency is assumed because the TOE of the Security Target (ST) is the Smart Card Platform (IC and OS) and the Java Card System, as expressed in [PP_JC_Closed]. Pre-issuance applets embedded in J-Safe ROM (Card Manager, J-Sign) are considered as part of TOE environment.

All threats, organizational security policies and assumptions defined in [PP_JC_Closed] apply to the TOE. Moreover, this Security Target defines:

- additional threats to the security problem definition, which are compatible with threats of the protection profile but they are more focused to generic attacks on the complete TOE, including the smartcard platform and Global Platform software. In particular, a new threat (T.LIFE_CYCLE) has been defined, while the threat T.PHYSICAL defined in the protection profile, has been refined. These threats do not introduce any contradiction with existing ones.
- additional organizational security policies (OSPs) to the security problem definition, which are compatible with organizational security policies of the protection profile and applicable to this TOE configuration. The new OSPs are: OSP.CARD_ADMINISTRATION_DISABLED, OSP.MANAGEMENT_OF_SECRETS and OSP.ROLES. These OSPs are not in contradiction with the [PP_JC_Closed] SPD.

All objectives defined in [PP_JC_Closed] have been included in this ST, thus guaranteeing the consistency of the statement of security objectives with the statement of security objectives in the PP for which conformance is being claimed.
Moreover, some objectives for environment in [PP_JC_Closed] have become objectives for J-Safe TOE due to inclusion of IC and OS in the ST scope. In particular, OE.SCP.IC, OE.SCP.RECOVERY, and OE.SCP.SUPPORT have been redefined respectively as *O.SCP.IC*, *O.SCP.RECOVERY*, and *O.SCP.SUPPORT* in the ST.

This ST also defines new objectives: O.SIDE_CHANNEL, O.LIFE_CYCLE*.* These objectives are not in contradiction with existing ones.

The list of SFRs used in the security target includes all the SFR described in [PP_JC_Closed].

In the SFR description, **this typeface** is used to highlight SFR assignments where they are expected by the Protection Profile.

This typeface is used to distinguish refinements that have been performed in this Security Target with respect to the ones already contained in [PP_JC_Closed].

This typeface is used to distinguish application notes, remarks, editorial changes and additional refinements added in this Security Target with respect to the ones already contained in [PP_JC_Closed].

Besides, with respect to [PP_JC_Closed], the following requirements have been refined:

PP SFR	Operation	ST SFR
fcs_ckm.1	Iteration	fcs_ckm.1/RSA fcs_ckm.1/EC fcs_ckm.1/DSA
fcs_ckm.2	Iteration	fcs_ckm.2/DES fcs_ckm.2/AES fcs_ckm.2/RSA_STD fcs_ckm.2/RSA_CRT fcs_ckm.2/EC fcs_ckm.2/DSA
fcs_ckm.3	Iteration	fcs_ckm.3/DES fcs_ckm.3/AES fcs_ckm.3/RSA_STD fcs_ckm.3/RSA_CRT fcs_ckm.3/EC fcs_ckm.3/DSA
fcs_cop.1	Iteration	fcs_cop.1/DES-TDES_Cipher fcs_cop.1/DES_MAC fcs_cop.1/AES_Cipher fcs_cop.1/AES_MAC fcs_cop.1/RSA_Cipher fcs_cop.1/RSA_Signature fcs_cop.1/EC Signature fcs_cop.1/SHA fcs_cop.1/ECDH_KeyExchange
fau_arp.1	Refinement	fau_arp.1/JCS

In order to fulfil the objective O.SIDE_CHANNEL and the 3 objectives related to SCP which is part of the TOE (*O.SCP.SUPPORT*, *O.SCP.IC*, *O.SCP.RECOVERY*), this ST adds several SFRs:

- some SFRs of the IC listed in the [STlite_SB23] have been included:

- fru_flt.2
- fpt_fls.1 (renamed in fpt_fls.1/SCP)
- fmt_lim.1
- fmt_lim.2
- fpt_php.3
- fdp_itt.1
- fpt_itt.1
- fdp_ifc.1
- fcs_rng.1

- additional SFRs (partially taken from [PP_ESforSSD]) have been explicitly added in:
  - fdp_acc.1/Atomicity
  - fdp_rol.1/Atomicity
  - fpt_fls.1/Operate
  - fpt_tst.1

In order to fulfil the objective O.SIDE_CHANNEL included in the TOE, the ST adds the following requirement:

- fpt_emsec.1

In order to fulfil the objective O.LIFE_CYCLE of the GP framework included in the TOE, the ST adds the following requirements:

- fdp_acc.1/GP_API
- fdp_acf.1/GP_API
- fmt_msa.1/GP_API
- fmt_msa.3/GP_API
- fmt_smr.1/GP_API
- fia_uid.1/GP_API

4.2.3 Statement of Compatibility concerning Composite Security Target

This is a Statement of Compatibility between this Composite ST and the Platform ST of the hardware and associated crypto-library [STlite_SB23].
The following mappings regarding SFRs, threats, assumptions, organizational security policies and objectives demonstrate the compatibility between the Composite Security Target and the Platform ST.

The following table lists the Platform Security Functionalities and classifies the Hardware SF as relevant or not relevant for the Composite ST.

Platform Security Functionality	Relevance	Composite ST TSF
TSF_INIT_A: Hardware initialisation & TOE attribute initialisation	RP_SF	SF.SmartCardPlatform
TSF_CONFIG_A: TOE configuration switching and control	IP_SF	
TSF_INT_A: TOE logical integrity	RP_SF	SF.SmartCardPlatform
TSF_TEST_A: Test of the TOE	IP_SF	
TSF_FWL_A: Memory Firewall	IP_SF	
TSF_PHT_A: Physical tampering protection	RP_SF	SF.SmartCardPlatform
TSF_ADMINIS_A: Security violation administrator	RP_SF	SF.SmartCardPlatform

TSF_OBS_A: Unobservability	RP_SF	SF.SecureManagement SF.SmartCardPlatform
TSF_SKCS_A: Symmetric Key Cryptography Support	RP_SF	SF.CryptoKey SF.CryptoOp SF.SmartCardPlatform
TSF_AKCS_A: Asymmetric Key Cryptography Support	RP_SF	SF.CryptoKey SF.CryptoOp SF.SmartCardPlatform
TSF_ALEAS_A: Unpredictable Number Generation Support	RP_SF	SF.CryptoKey SF.CryptoOp SF.SmartCardPlatform

Table 1 - Platform Security Functionality relevance for the composite TOE

NOTE: TSF_CONFIG_A and TSF_TEST_A are considered not relevant for the composite TOE because they are functionalities used only by the IC Software.

The table below shows the mapping between the Platform SFRs and the Composite ST SFRs. Both the relevant and the irrelevant SFRs are listed.

Platform SFRs	Composite ST SFRs
FRU_FLT.2	FRU_FLT.2
FPT_FLS.1	FPT_FLS.1/SCP
FMT_LIM.1	FMT_LIM.1
FMT_LIM.2	FMT_LIM.2
FAU_SAS.1	Not relevant / Not used
FPT_PHP.3	FPT_PHP.3/SCP
FDP_ITT.1	FDP_ITT.1/SCP
FPT_ITT.1	FPT_ITT.1/SCP
FDP_IFC.1	FDP_IFC.1/SCP
FCS_RNG.1	FCS_RNG.1
FCS_COP.1	Contributes to implementation of FCS_COP.1 of JCS: FCS_COP.1/DES-TDES_Cipher FCS_COP.1/DES_MAC FCS_COP.1/AES_Cipher FCS_COP.1/AES_MAC FCS_COP.1/RSA_Cipher FCS_COP.1/RSA_Signature FCS_COP.1/EC_Signature FCS_COP.1/SHA FCS_COP.1/ECDH_KeyExchange
FCS_CKM.1	Contributes to implementation of FCS_CKM.1 of JCS: FCS_CKM.1/RSA FCS_CKM.1/EC FCS_CKM.1/DSA
FDP_ACC.2	Not relevant / Not used
FDP_ACF.1	Not relevant / Not used
FMT_MSA.3	Not relevant / Not used
FMT_MSA.1	Not relevant / Not used

Table 2 - Platform SFRs VS Composite TOE SFRs

There is no conflict between policies of the Composite ST and the Platform ST:

Platform OSPs	Platform Objective	Composite ST Policies
P.Process-TOE	BSI.O.Identification	
AUG1.P.Add-Functions	AUG1.O.Add-Functions	
		Additional Policies OSP.MANAGEMENT_OF_SECRETS OSP.CARD_ADMINISTRATION_DISABLED OSP.VERIFICATION OSP.ROLES

Table 3 - Platform OSPs VS Composite TOE OSPs

There is no conflict between assumptions of the Composite ST and the Platform ST:

Platform Assumptions	Assumption Classification	Composite ST Assumptions
A.Plat-Appl	IrPA	
A.Resp-Appl	CfPA	
A.Process-Sec-IC	IrPA	
		Additional Assumptions A.VERIFICATION A.NO-DELETION A.NO-INSTALL

Table 4 - Platform Assumptions VS Composite TOE Assumptions

There is no conflict between security objectives of the Composite ST and the Platform ST:

Platform Objectives	Composite ST Objectives	Remarks
O.Identification	Not relevant	
O.Leak-Inherent **O.Leak-Forced**	*O.SCP.IC* *O.SCP.SUPPORT* O.SIDE_CHANNEL	
O.Phys-Probing **O.Phys-Manipulation**	*O.SCP.IC*	
O.Malfunction	*O.SCP.RECOVERY* *O.SCP.IC* *O.SCP.SUPPORT*	
O.Abuse-Func	Not relevant	
O.RND	*O.SCP.IC*	
AUG1.O.Add-Functions	*O.SCP.SUPPORT* O.CIPHER	
AUG4.O.Mem Access	Not relevant	
	Additional Objectives O.SID O.FIREWALL O.GLOBAL_ARRAYS_CONFID O.GLOBAL_ARRAYS_INTEG O.NATIVE O.RESOURCES O.REALLOCATION O.ALARM O.KEY_MNGT O.PIN_MNGT O.TRANSACTION O.OBJ-DELETION O.OPERATE O.LIFE_CYCLE O.ROLES	

Table 5 - Platform Objectives VS Composite TOE Objectives

There is no conflict between security objectives for the environment of the Composite ST and the Platform ST:

Platform OE	Composite ST Objectives for the Environment
OE.Plat-Appl	Not relevant
OE.Resp-Appl	Not relevant
OE.Process-Sec-IC	OE. MANAGEMENT_OF_SECRETS
	Additional Objectives for the Environment OE.NO-DELETION OE.NO-INSTALL OE.VERIFICATION OE.CARD_MANAGEMENT

Table 6 - Platform OEs VS Composite TOE OEs

4.3 Security Problem Definition (ASE_SPD)

4.3.1 Security Aspects

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

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

TSFs must be continuously active in one way or another; this is called "OPERATE". The Protection Profile may include an assumption, called "A.OPERATE", stating that it is assumed that the TOE ensures continued correct operation of its security functions, and so on. However, it may also include a threat, called "T.OPERATE", to be interpreted as the negation of the statement .OPERATE. In this example, this amounts to stating that an attacker may try to circumvent some specific TSF by temporarily shutting it down. The use of "OPERATE" is intended to ease the understanding of this document.

This section presents security aspects that will be used in the remainder of this document. Some being quite general, we give further details, which are numbered for easier cross-reference within the document. For instance, the two parts of #.OPERATE, when instantiated with an objective "O.OPERATE", may be met by separate SFRs in the rationale. The numbering then adds further details on the relationship between the objective and those SFRs.

4.3.1.1 CONFIDENTIALITY

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

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

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

4.3.1.2 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.

#.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.

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

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

4.3.1.3 UNAUTHORIZED EXECUTIONS

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

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

#.FIREWALL The Firewall shall ensure controlled sharing of class instances1, and isolation of their data and code between packages (that is, controlled execution contexts) as well as between packages and the JCRE context. An applet shall not read, write or compare a piece of data belonging to an applet that is not in the same context, or execute one of the methods of an applet in another context without its authorization.

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

4.3.1.4 BYTECODE VERIFICATION

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

CAP FILE VERIFICATION

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

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

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

on the card capabilities, in order to ensure that each bytecode is valid at execution time. This Protection Profile assumes bytecode verification is performed off-card.

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

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. On-card bytecode verifier is out of the scope of this Protection Profile.

LINKING AND VERIFICATION

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

4.3.1.5 CARD MANAGEMENT

#.CARD-MANAGEMENT The card manager shall implement the card issuer’s policy on the card.

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

#.SID (1) Users and subjects of the TOE must be identified. (2) The identity of sensitive users and subjects associated with administrative and privileged roles must be particularly protected; this concerns the Java Card RE, 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.x). A change of identity, especially standing for an administrative role (like an applet impersonating the Java Card RE), is a severe violation of the Security Functional Requirements (SFR). Selection controls the access to any data exchange between the TOE and the CAD and therefore, must be protected as well. Any exchange of data through the APDU buffer (which can be accessed by any applet) can lead to disclosure of keys, application code or data, and so on.

#.OBJ-DELETION (1) Deallocation of objects should not introduce security holes in the form of references pointing to memory zones that are not longer in use, or have been reused for other purposes. Deletion of collection of objects should not be maliciously used to circumvent the TSFs. (2) Erasure, if deemed successful, shall ensure that the deleted class instance is no longer accessible.

4.3.1.6 SERVICES

#.ALARM The TOE shall provide appropriate feedback upon detection of a potential security violation. This particularly concerns the type errors detected by the bytecode verifier, the security exceptions thrown by the Java Card VM, 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 SFRs. Then: (1) After a power loss, RF signal 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 SFRs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the Java Card 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. Finally, it is required that (7) the IC is designed in accordance with a well-defined set of policies and standards (for instance, those specified in PP_0035), 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 jeopardize the execution of the user applications. The transaction status at the beginning of an applet session must be closed (no pending updates).

4.3.2 Assets

This chapter introduces the assets to be protected, the users of the TOE and their software counterparts.

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

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

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

4.3.2.1 User Data

D.APP_CODE
The code of the applets and libraries loaded on the card.
To be protected from unauthorized modification.

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

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

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

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

4.3.2.2 TSF Data

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

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

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

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

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

D.EMBEDDED_SW_CODE
The code of the software embedded in the product. The asset stands for the native OS code.
Note: This asset is related to native OS code only. The JCS code is not included in this asset because it is covered by D.JCS_CODE
To be protected from unauthorized disclosure and modification.

D.EMBEDDED_SW_DATA
The sensitive TSF data processed or stored by the product. The asset stands for the data of the native OS.

Note: This asset is related to native OS data only. The JCS data is not included in this asset because it is covered by D.JCS_DATA

To be protected from unauthorized disclosure and modification.

D.GP_CODE
The code of the GlobalPlatform framework.
To be protected from unauthorized disclosure and modification.

D.GP_SENSITIVE_DATA
Application Privilege, Application Life Cycle State, Card Life Cycle State.
To be protected from unauthorized modification.

4.3.3 Threats

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

4.3.3.1 Confidentiality

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

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

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

4.3.3.2 Integrity

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

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

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

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

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

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

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

4.3.3.3 Identity usurpation

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

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

4.3.3.4 Unauthorized execution

T.EXE-CODE.1
An applet performs an unauthorized execution of a method. See #.EXE-JCS-CODE and #.EXE-APPLI-CODE for details.
Directly threatened asset(s): D.APP_CODE.

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

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

<u>Refinement:</u>
This threat also addresses several attack methods to modify the product behaviour:
- *unauthorized use of commands (one or many incorrect commands, undefined commands, hidden commands)*
- *buffer overflow attacks (overwriting buffer content to modify execution contexts or gaining system privileges).*

Additional threatened asset(s): D.EMBEDDED_SW_CODE, D.GP_CODE, since TOE defined in this ST includes also SCP platform and GP framework.

4.3.3.5 Denial of service

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

4.3.3.1 Services

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

4.3.3.2 Miscellaneous

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

Refinement:
This threat also addresses leakage of information that may occur during TOE usage through:
- *Emanations,*
- *Variations in power consumption,*
- *I/O characteristics,*
- *Clock frequency,*
- *Changes in processing time*

T.LIFE_CYCLE
An attacker accesses to product functionalities outside of their expected availability range thus violating irreversible life cycle phases of the product (for instance, an attacker downloads, install, or delete applications available on the product at post-issuance).
Threatened asset(s): D.EMBEDDED_SW_CODE, D.EMBEDDED_SW_DATA, D.GP_SENSITIVE_DATA, D.GP_CODE, D.APP_CODE.

4.3.4 Organizational Security Policies

OSP.VERIFICATION
This policy shall ensure the consistency between the export files used in the verification and those used for installing the verified file. The policy must also ensure that no modification of the file is performed in between its verification and the signing by the verification authority. See #.VERIFICATION for details.

OSP.MANAGEMENT_OF_SECRETS
Management of secret data (e.g. generation, storage, distribution, destruction, loading into the product of cryptographic private keys, symmetric keys, user authentication data) performed outside the product on behalf of the TOE or Product Manufacturer shall comply with security organisational policies that enforce integrity and confidentiality of these data.
Secret data shared with the user of the product shall be exchanged through trusted channels that protect the data against unauthorised disclosure and modification and allow detecting potential security violations.

OSP.ROLES
The TOE shall recognize the following roles associated with:
- Applications

OSP.CARD_ADMINISTRATION_DISABLED
Card Content Management Functions (CCMFs) shall not be available after TOE delivery.

4.3.5 Assumptions

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

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.

A.NO-DELETION
No deletion of installed applets (or packages) is possible.
NOTE: Refers to TOE usage phases.

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 for details.
NOTE: Refers to TOE usage phases.

4.4 Security objectives (ASE_OBJ)

4.4.1 Security objectives for the TOE

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

4.4.1.1 Identification

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

O.ROLES
The TOE shall recognize the following roles associated with:

- Applications

4.4.1.2 Execution

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

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

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

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

O.OPERATE
The TOE shall ensure continued correct operation of its security functions, prevent the unauthorised use of commands and ensure that patches to the code are not bypassed. See #.OPERATE for details.

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

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

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

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

- It is required that the IC is designed in accordance with a well-defined set of policies and standards (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.

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

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

4.4.1.3 Services

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

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

O.KEY-MNGT
The TOE shall provide a means to securely manage cryptographic keys. This concerns the correct generation, distribution, access and destruction of cryptographic keys. See #.KEY-MNGT.

O.PIN-MNGT
The TOE shall provide a means to securely manage PIN objects. See #.PIN-MNGT for details.
Application Note:
PIN objects may play key roles in the security architecture of client applications. The way they are stored and managed in the memory of the smart card must be carefully considered, and this applies to the whole object rather than the sole value of the PIN. For instance, the try counter's value is as sensitive as that of the PIN.

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

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 are also be present on the card and made available to applets; those are built on top of the Java Card API or independently. These proprietary libraries will be evaluated together with the TOE.

4.4.1.1 Object Deletion

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

4.4.1.2 Embedded Software

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 software 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.LIFE_CYCLE
The TOE shall manage its own life cycle states and support user applications in the management of their life cycle state, including reversible and irreversible transitions between them. The TOE shall reject operations unexpected in its current life cycle.

4.4.2 Security objectives for the environment

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

OE.CARD_MANAGEMENT
The card manager shall control the access to card management functions such as the installation, update or deletion of applets. It shall also implement the card issuer's policy on the card.
The Card Manager (also called Issuer Security Domain, ISD) is an application with specific rights, which is responsible for the administration of the smart card. This component will in practice be tightly connected with the rest of the TOE, which in turn shall very likely rely on the card manager for the effective enforcing of some of its security functions.
The Card Manager shall be in charge of the life cycle of the whole card, as well as that of the installed applications (applets). The card manager shall prevent that card content management

(loading, installation, deletion) is carried out, for instance, at invalid states of the card or by non-authorized actors. It shall also enforce security policies established by the card issuer.

OE.NO-DELETION
No installed applets (or packages) shall be deleted from the card.

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

OE.VERIFICATION
All the bytecodes shall be verified at least once, before the loading, before the installation or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. See #.VERIFICATION for details.
Additionally, the applet shall follow all the recommendations, if any, mandated in the platform guidance for maintaining the isolation property of the platform.

Application Note: Constraints to maintain the isolation property of the platform are provided by the platform developer in application development guidance. The constraints apply to all application code loaded in the platform. *On current TOE, package loading shall only occur in pre-issuance phase and in such conditions, bytecode verification shall be performed before loading the package.*

NOTE: 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 Closed configuration corresponds to that of a closed card (no code can be loaded or deleted once the card has been issued).

OE.MANAGEMENT_OF_SECRETS
The secret User or TSF data managed outside the TOE shall be protected against unauthorised disclosure and modification.

4.4.3 Security objectives rationale

4.4.3.1 SPD and Security Objectives Relation

T.CONFID-APPLI-DATA
This threat is countered by the security objective for the operational environment regarding bytecode verification (OE.VERIFICATION). It is also covered by the isolation commitments stated in the (O.FIREWALL) objective. It relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode respectively.
The objectives *O.SCP.RECOVERY* and *O.SCP.SUPPORT* are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.
As applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys and PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) shall contribute in covering this threat by controlling the sharing of the global PIN between the applets.

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

T.CONFID-JCS-CODE
This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of those instructions enables reading a piece of code, no Java Card applet can therefore be executed to disclose a piece of code. Native applications are also harmless because of the objective (O.NATIVE), so no application can be run to disclose a piece of code.
The (#.VERIFICATION) security aspect is addressed in this PP by the objective for the environment OE.VERIFICATION.
The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

T.CONFID-JCS-DATA
This threat is covered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) security objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objectives *O.SCP.RECOVERY* and *O.SCP.SUPPORT* are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

T.INTEG-APPLI-CODE
This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective (O.NATIVE), so no application can be run to modify a piece of code.
The (#.VERIFICATION) security aspect is addressed in this configuration by the objective for the environment OE.VERIFICATION.
The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling *the* access to card management functions and by checking the bytecode, respectively.

T.INTEG-APPLI-CODE.LOAD
By controlling the access to card management functions such as the installation, update or deletion of applets the objective OE.CARD-MANAGEMENT contributes to cover this threat.

T.INTEG-APPLI-DATA
This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

The objectives *O.SCP.RECOVERY* and *O.SCP.SUPPORT* are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.
Concerning the confidentiality and integrity of application sensitive data, as applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys and PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) is also concerned.
Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The integrity of the information stored in that buffer is ensured by the (O.GLOBAL_ARRAYS_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.

T.INTEG-APPLI-DATA.LOAD
By controlling the access to card management functions such as the installation, update or deletion of applets the objective OE.CARD-MANAGEMENT contributes to cover this threat.

T.INTEG-JCS-CODE
This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective O.NATIVE, so no application can be run to modify a piece of code.
The (#.VERIFICATION) security aspect is addressed in this configuration by the objective for the environment OE.VERIFICATION.
The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

T.INTEG-JCS-DATA
This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives OE.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objectives *O.SCP.RECOVERY* and *O.SCP.SUPPORT* are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

T.SID.1
As impersonation is usually the result of successfully disclosing and modifying some assets, this threat is mainly countered by the objectives concerning the isolation of application data (like PINs), ensured by the (O.FIREWALL). Uniqueness of subject-identity (O.SID) also participates to face this threat. It should be noticed that the AIDs, which are used for applet identification, are TSF data.
The installation parameters of an applet (like its name) are loaded into a global array that is also shared by all the applications. The disclosure of those parameters (which could be used to impersonate the applet) is countered by the objective (O.GLOBAL_ARRAYS_CONFID) and (O.GLOBAL_ARRAYS_INTEG).
The objective OE.CARD-MANAGEMENT contributes, by preventing usurpation of identity resulting from a malicious installation of an applet on the card, to counter this threat.

T.SID.2
This is covered by integrity of TSF data, subject-identification (O.SID), the firewall (O.FIREWALL) and its good working order (O.OPERATE).
The objectives *O.SCP.RECOVERY* and *O.SCP.SUPPORT* are intended to support the O.OPERATE objective of the TOE, so they are indirectly related to the threats that this latter objective contributes to counter.

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

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

T.NATIVE
This threat is countered by O.NATIVE which ensures that a Java Card applet can only access native methods indirectly that is, through a secure API. In addition to this, the bytecode verifier also prevents the program counter of an applet to jump into a piece of native code by confining the control flow to the currently executed method (OE.VERIFICATION).
Refinement:
In addition, O.OPERATE guarantees that only authorized use of commands is allowed thus avoiding that normal product behaviour is modified.

T.RESOURCES
This threat is directly countered by objectives on resource-management (O.RESOURCES) for runtime purposes and good working order (O.OPERATE) in a general manner.
It should be noticed that, for what relates to CPU usage, the Java Card platform is single-threaded and it is possible for an ill-formed application (either native or not) to monopolize the CPU. However, a smart card can be physically interrupted (card removal or hardware reset) and most CADs implement a timeout policy that prevent them from being blocked should a card fails to answer. That point is out of scope of this Protection Profile, though.
Finally, the objectives *O.SCP.RECOVERY* and *O.SCP.SUPPORT* are intended to support the O.OPERATE and O.RESOURCES objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

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

T.PHYSICAL [Editorially Refined since SCP is part of the TOE]
Covered by O.SCP.IC,.that ensures protection against any kind of physical attack, and by O.SIDE_CHANNEL that provides protection against disclosure of confidential data that could happen by measurement and analysis of signals' shape, amplitude and time.
O.ALARM gives feedback information when a potential security violation is detected.
Finally O.SCP.SUPPORT gives support to the other objectives ensuring low-level protection against TSF alteration.

T.LIFE_CYCLE
O.SCP.SUPPORT prevents the TSFs from being bypassed or altered including violation of irreversible life-cycle states, O.LIFE_CYCLE requires the control of the IC and TOE life cycles, respectively, to prevent abuse of functionality by physical and logical means. The fulfilment of these two objectives allows removing the threat.

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

OSP.MANAGEMENT_OF_SECRETS
OE.Management_of_Secrets directly covers the security policy.

OSP.ROLES
It is entirely addressed by O.ROLES.

OSP.CARD_ADMINISTRATION_DISABLED
It is entirely addressed by OE.CARD_MANAGEMENT.

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

A.NO-DELETION
The assumption A.NO-DELETION is upheld by the environmental objective OE.NO-DELETION which guarantees that no installed applets (or packages) shall be deleted from the card. The environmental objective OE.CARD-MANAGEMENT also upholds this assumption by controlling the access to card management functions such as applets deletion.

A.NO-INSTALL
This assumption is upheld by the environmental objective OE.NO-INSTALL which imposes that no post-issuance installation of applets is permitted. The objective OE.CARD-MANAGEMENT contributes in upholding this assumption by controlling the access to card management functions such as the installation of applets.

4.4.3.2 SPD and Security Objectives Rationale

Threat	Objectives
T.CONFID-APPLI-DATA	O.SID, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.OPERATE, O.REALLOCATION, *O.SCP.RECOVERY* *O.SCP.SUPPORT*, O.ALARM, O.CIPHER, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION OE.CARD_MANAGEMENT, OE.VERIFICATION
T.CONFID-JCS-CODE	O.NATIVE , OE.CARD_MANAGEMENT OE.VERIFICATION
T.CONFID-JCS-DATA	O.SID, O.FIREWALL, O.OPERATE, *O.SCP.RECOVERY, O.SCP.SUPPORT*, O.ALARM OE.CARD_MANAGEMENT, OE.VERIFICATION
T.INTEG-APPLI-CODE	O.NATIVE, OE.CARD_MANAGEMENT OE.VERIFICATION
T.INTEG-APPLI-CODE.LOAD	OE.CARD_MANAGEMENT
T.INTEG-APPLI-DATA	O.SID , O.FIREWALL, O.GLOBAL_ARRAYS_INTEG O.OPERATE, O.REALLOCATION, *O.SCP.RECOVERY* *O.SCP.SUPPORT*, O.ALARM, O.CIPHER O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION OE.CARD_MANAGEMENT, OE.VERIFICATION

T.INTEG-APPLI-DATA.LOAD	OE.CARD_MANAGEMENT
T.INTEG-JCS-CODE	O.NATIVE, OE.CARD_MANAGEMENT, OE.VERIFICATION
T.INTEG-JCS-DATA	O.SID, O.FIREWALL, O.OPERATE, *O.SCP.RECOVERY, O.SCP.SUPPORT*, O.ALARM, OE.CARD_MANAGEMENT, OE.VERIFICATION
T.SID.1	O.SID, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID O.GLOBAL_ARRAYS_INTEG, OE.CARD_MANAGEMENT
T.SID.2	O.SID, O.FIREWALL, O.OPERATE, *O.SCP.RECOVERY, O.SCP.SUPPORT*
T.EXE-CODE.1	O.FIREWALL, OE.VERIFICATION
T.EXE-CODE.2	OE.VERIFICATION
T.NATIVE	O.NATIVE, OE.VERIFICATION, O.OPERATE
T.RESOURCES	O.OPERATE, O.RESOURCES, *O.SCP.RECOVERY O.SCP.SUPPORT*
T.OBJ-DELETION	O.OBJ-DELETION
T.PHYSICAL **T.LIFE_CYCLE**	*O.SCP.IC, O.SCP.SUPPORT*, O.ALARM, O.SIDE_CHANNEL *O.SCP.SUPPORT*, O.LIFE_CYCLE
OSP.MANAGEMENT_OF_SECRETS	OE.MANAGEMENT_OF_SECRETS
OSP.ROLES	O.ROLES
OSP.CARD_ADMINISTRATION_DISABLED	OE.CARD_MANAGEMENT
OSP.VERIFICATION	OE.VERIFICATION
A.VERIFICATION	OE.VERIFICATION
A.NO-DELETION	OE.NO-DELETION , OE.CARD_MANAGEMENT

A.NO-INSTALL	OE.NO-INSTALL, OE.CARD_MANAGEMENT

Table 7 – SPD vs. Objectives Rationale

Objectives	Threats
O.SID	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.1, T.SID.2
O.ROLES	OSP.ROLES
O.FIREWALL	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.1, T.SID.2 T.EXE-CODE.1
O.GLOBAL_ARRAYS_CONFID	T.CONFID-APPLI-DATA, T.SID.1
O.GLOBAL_ARRAYS_INTEG	T.INTEG-APPLI-DATA, T.SID.1
O.NATIVE	T.CONFID-JCS-CODE, T.INTEG-APPLI-CODE, T.INTEG-JCS-CODE T.NATIVE
O.OPERATE	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2 T.RESOURCES, T.PHYSICAL
O.RESOURCES	T.RESOURCES
O.REALLOCATION	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.SCP.RECOVERY	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES, T.LEAKAGE, T.LIFE_CYCLE
O.SCP.IC	T.PHYSICAL
O.SCP.SUPPORT	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES, T.LIFE_CYCLE, T.PHYSICAL
O.ALARM	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.PHYSICAL
O.CIPHER	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.KEY-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA

O.PIN-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.TRANSACTION	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.OBJ-DELETION	T.OBJ-DELETION
O.SIDE_CHANNEL	T.PHYSICAL
O.LIFE_CYCLE	T.LIFE_CYCLE
OE.CARD_MANAGEMENT	T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA, T.INTEG-APPLI-DATA.LOAD, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.SID.1, OSP.CARD_ADMINISTRATION_DISABLED
OE.NO-DELETION	A.NO-DELETION
OE.NO-INSTALL	A.NO-INSTALL
OE.VERIFICATION	T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-DATA, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.EXE-CODE.1, T.EXE-CODE.2, T.NATIVE, OSP.VERIFICATION
OE.MANAGEMENT_OF_SECRETS	OSP.MANAGEMENT_OF_SECRETS

Table 8 - Objectives vs. SPD Rationale

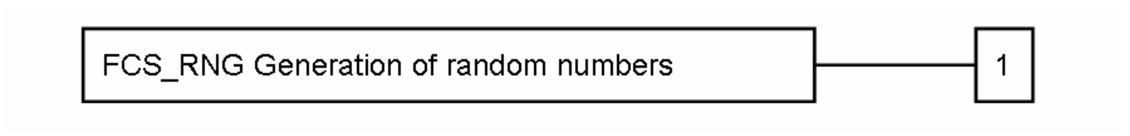
4.5 Extended Components Definition (ASE_ECD)

4.5.1 Definition of Family FCS_RNG

Section extracted from the Smartcard IC Platform Protection profile PP_0035 (cert. BSI-PP-0035)

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

Component leveling:



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

Management: FCS_RNG.1
There are no management activities foreseen.

Audit: FCS_RNG.1
There are no actions defined to be auditable.

FCS_RNG.1 Random number generation.

Hierarchical to: No other components.

Dependencies: No dependencies.

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

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

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

4.5.2 Definition of Family FMT_LIM

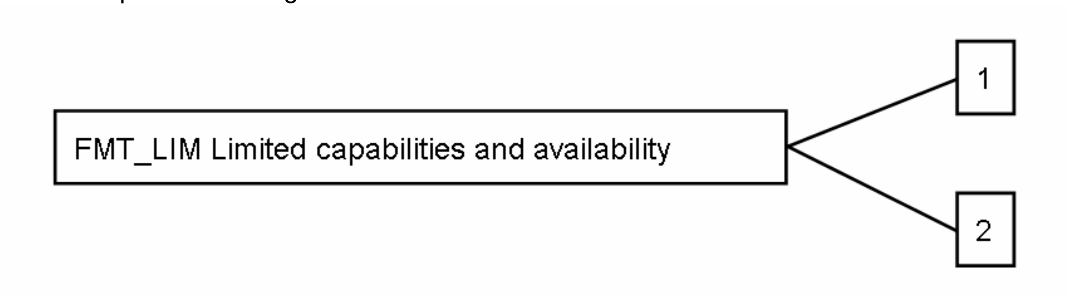
Section extracted from the Smartcard IC Platform Protection profile PP_0035 (cert. BSI-PP-0035)

To define the IT security functional requirements of the TOE an additional family (FMT_LIM) of the Class FMT (Security Management) is defined here. The 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 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 behavior: 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 component 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.

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

FMT_LIM.1 Limited capabilities.

Hierarchical to: No other components.

Dependencies: No dependencies.

FMT_LIM.1.1 The TSF shall be designed and implemented in a manner that limits its 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.

Dependencies: No dependencies.

FMT_LIM.2.1 The TSF shall be designed in a manner that limits its 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 (limitation of capabilities and limitation of availability) which together shall provide protection in order to enforce the policy. This also allows that

i. 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

ii. 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.

4.5.3 Definition of Family FPT_EMSEC

Section extracted from the Protection Profile PP_ICAO_BAC *(cert. BSI-CC-PP-0055).*

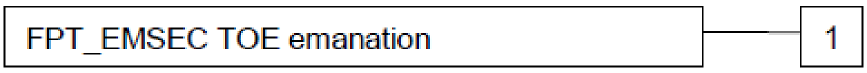
The sensitive family FPT_EMSEC (TOE Emanation) of the Class FPT (Protection of the TSF) is defined here to describe the IT security functional requirements of the TOE. The TOE shall prevent attacks against the TOE and other secret data where the attack is based on external observable physical phenomena 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 CC part 2 CC2.

The family "TOE Emanation (FPT_EMSEC)" is specified as follows.

FPT_EMSEC TOE Emanation

Family behavior: This family defines requirements to mitigate intelligible emanations.

Component leveling:



FPT_EMSEC.1 TOE emanation has two constituents:

FPT_EMSEC.1.1 TOE Limit of Emissions requires to not emit intelligible emissions enabling access to TSF data or user data.

FPT_EMSEC.1.2 Interface Emanation requires to 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.

The TOE Functional Requirement "TOE emanation (FPT_EMSEC.1)" is specified as follows.

FPT_EMSEC.1 TOE emanation

Hierarchical to: No other components.

Dependencies: No dependencies.

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*].

4.6 Security requirements (ASE_REQ)

4.6.1 Security functional requirements for the TOE (SFRS)

This section states the security functional requirements for the Java Card System - Closed configuration. For readability and for compatibility with the original Java Card System Protection Profile Collection - Standard 2.2 Configuration [PP/0305] ([PP-JCS-1.0]), requirements are arranged into groups.

The SFRs refer to all potentially applicable subjects, objects, information, operations and security attributes, including JCRMI related entities which are optional. Since the TOE does not provide JCRMI functionality, the associated entities and their corresponding requirements have been ignored in this ST.

Group	Description
Core with Logical Channels (*CoreG_LC*)	The CoreG_LC contains the requirements concerning the runtime environment of the Java Card System implementing logical channels. This includes the firewall policy and the requirements related to the Java Card API. Logical channels are a Java Card specification version 2.2 feature. This group is the union of requirements from the Core (*CoreG*) and the Logical channels (*LCG*) groups defined in [PP/0305] (cf. Java Card System Protection Profile Collection [PP-JCS-1.0]).
Remote Method Invocation (RMI)	The RMIG contains the security requirements for the remote method invocation feature, which provides a new protocol of communication between the terminal and the applets. This feature was introduced in Java Card specification version 2.2 and became optional in Java Card specification version 3 Classic Edition. NOTE: <u>This group of SFRs does not apply to current TOE since it doesn't provide JCRMI functionality</u>.
Object Deletion (*ODELG*)	The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism. This is a Java Card specification version 2.2 feature.

Subjects are active components of the TOE that (essentially) act on the behalf of users. The users of the TOE include people or institutions (like the applet developer, the card issuer, the verification authority), hardware (like the CAD where the card is inserted or the PCD) and software components (like the application packages installed on the card). Some of the users may just be aliases for other users. For instance, the verification authority in charge of the bytecode verification of the applications may be just an alias for the card issuer.

<u>Note</u>: with respect to [PP_JC_Closed], additional subjects, objects, security attributes and operations have been added, to take into account parts of the TOE that in [PP_JC_Closed] were part of the TOE Environment. Subjects, objects, security attributes and operations are described in the following tables. The last column in each table indicates whether they are referred to COREG_LC, ODELG, or GP_API policy.

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

Subject	Description	Related Policy
S.JCRE	The Java Card RE is responsible on behalf of the card issuer of the bytecode execution and runtime environment functionalities. It is the process that manages applet selection and de-selection, along with the delivery of APDUs from and to the smart card device. This subject is unique.	COREG_LC
S.JCVM	The Java Card VM is the bytecode interpreter. This subject dynamically enforces the firewall, that is, at runtime.	COREG_LC
S.LOCAL	Operand stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.	COREG_LC
S.MEMBER	Any object's field, static field or array position.	COREG_LC
S.PACKAGE	A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of Java Card technology, it defines either a user library, or one or several applets	COREG_LC
S.APPLICATION	A Java Card Application	GP_API

Table 9 - Subjects of the TOE

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

Object	Description	Related Policy
O.APPLET	Any installed applet, its code and data.	COREG_LC
O.JAVAOBJECT	Java class instance or array. It should be noticed that KEYS, PIN, arrays and applet instances are specific objects in the Java programming language.	COREG_LC
O.GP_REGISTRY	Global Platform Registry	GP_API

Table 10 - Objects of the TOE

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

Information	Description	Related Policy
I.DATA	JCVM Reference Data: objectref addresses of APDU buffer, JCRE-owned instances of APDU class and byte array for install method.	COREG_LC

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

Security Attribute	Description	Related Policy
Active Applets	The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels.	COREG_LC
Applet Selection Status	Selected" or "Deselected".	COREG_LC

Applet's version number	The version number of an applet (package) indicated in the export file.	COREG_LC
Context	Package AID or "Java Card RE".	COREG_LC
Currently Active Context	Package AID or "Java Card RE".	COREG_LC
LC Selection Status	Multiselectable, Non-multiselectable or "None".	COREG_LC
LifeTime	CLEAR_ON_DESELECT or PERSISTENT (*).	COREG_LC
Package AID	The AID of each package indicated in the export file.	COREG_LC
Registered applet AID	The AID of the applet instance registered on the card.	COREG_LC
Selected Applet Context	Package AID or "None".	COREG_LC
Sharing	Standards, SIO, Java Card RE entry point, or global array.	COREG_LC
Card Life Cycle State	Card life-cycle state as defined by GlobalPlatform Card Specification ([GP211])	GPAPI
AID	The Application AID	GP_API
ApplicationPrivilege	The Application Privilege	GP_API
ApplicationLifeCycleState	The Application Life Cycle State	GP_API
ApplicationAID	The Application AID stored in the GP Registry	GP_API

Table 11 - Security Attributes and related description

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

Operations (prefixed with "OP") are described in the following table. Each operation has 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.

Operation	Description	Related Policy
OP.ARRAY_ACCESS(O.JAVA OBJECT, field)	Read/Write an array component.	COREG_LC
OP.CREATE(Sharing, LifeTime) (*)	Creation of an object (new or makeTransient call).	COREG_LC
OP.INSTANCE_FIELD(O.JAVA OBJECT, field)	Read/Write a field of an instance of a class in the Java programming language.	COREG_LC
OP.INVK_VIRTUAL(O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object).	COREG_LC
OP.INVK_INTERFACE(O.JAVA OBJECT, method, arg1,...)	Invoke an interface method.	COREG_LC
OP.JAVA(...)	Any access in the sense of JCRE3, §6.2.8. It stands for one of the operations: **OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW, OP.TYPE_ACCESS.**	COREG_LC
OP.PUT(S1,S2,I)	Transfer a piece of information I from S1 to S2.	COREG_LC
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see JCRE3, §6.2.8.7).	COREG_LC
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	Invoke checkcast or instanceof on an object in order to access to classes (standard or shareable interfaces objects).	COREG_LC
OP.MANAGE_APP_LIFE_CYCLE_STATE	Return or change the Application Life Cycle state	GP_API
OP.GET_CARD_LIFE_CYCLE_STATE	Return the current Card Life Cycle state	GP_API
OP.CARD_LOCK	Lock the card	GP_API
OP.CARD_TERMINATE	Terminate the card	GP_API

Table 12 - Operations and related description

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

4.6.1.1 COREG_LC Security Functional Requirements

The CoreG_LC contains the requirements concerning the runtime environment of the Java Card System implementing logical channels. This includes the firewall policy and the requirements related to the Java Card API. Logical channels are a Java Card specification version 2.2 feature.

Firewall Policy

fdp_acc.2/FIREWALL - Complete access control

fdp_acc.2.1/FIREWALL: The TSF shall enforce the **FIREWALL access control SFP** on **S.PACKAGE, S.JCRE, S.JCVM, O.JAVAOBJECT** and all operations among subjects and objects covered by the SFP. The operations involved in the policy are:
- OP.CREATE
- OP.INVK_INTERFACE
- OP.INVK_VIRTUAL
- OP.JAVA
- OP.THROW
- OP.TYPE_ACCESS

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

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

fdp_acf.1/FIREWALL - Security attribute based access control

fdp_acf.1.1/FIREWALL: The TSF shall enforce the **FIREWALL access control SFP** to objects based on the following:
- **S.PACKAGE**: LC Selection Status
- **S.JCVM**: Active Applets, Currently Active Context
- **S.JCRE**: Selected Applet Context
- **O.JAVAOBJECT**: Sharing, Context, LifeTime.

fdp_acf.1.2/FIREWALL: The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:
- **R.JAVA.1 (**JCRE3**, sect.6.2.8): S.PACKAGE may freely perform OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW or OP.TYPE_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".**

- **R.JAVA.2 (**JCRE3**, sect.6.2.8): S.PACKAGE may freely perform OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE or OP.THROW upon any O.JAVAOBJECT whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if O.JAVAOBJECT's Context attribute has the same value as the active context.**

- **R.JAVA.3 (**JCRE3**, sect.6.2.8.10): S.PACKAGE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT whose Sharing attribute has value "SIO" only if O.JAVAOBJECT is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.**

- **R.JAVA.4 (**JCRE3**, sect.6.2.8.6): S.PACKAGE may perform OP.INVK_INTERFACE**

upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if the invoked interface method extends the Shareable interface and one of the following conditions applies:

- **The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable",**

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

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

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

- **The subject S.JCRE can freely perform OP.JAVA(") and OP.CREATE, with the exception given in FDP_ACF.1.4/FIREWALL, provided it is the Currently Active Context.**

- **The only means that the subject S.JCVM shall provide for an application to execute native code is the invocation of a Java Card API method (through OP.INVK_INTERFACE or OP.INVK_VIRTUAL).**

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

- **Any subject with OP.JAVA upon an O.JAVAOBJECT whose LifeTime attribute has value "CLEAR_ON_DESELECT" if O.JAVAOBJECT's Context attribute is not the same as the Selected Applet Context.**

- **Any subject attempting to create an object by the means of OP.CREATE and a "CLEAR_ON_DESELECT" LifeTime parameter if the active context is not the same as the Selected Applet Context.**

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

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

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

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

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

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

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

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

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

fdp_ifc.1/JCVM - Subset information flow control

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

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

fdp_iff.1/JCVM - Simple security attributes

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

- **S.JCVM: Currently Active Context.**

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

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

fdp_iff.1.3/JCVM: The TSF shall enforce the **following additional information flow control SFP rules: none**.

fdp_iff.1.4/JCVM: The TSF shall explicitly authorise 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 Java Card RE-owned objects references is runtime-enforced (JCRE3, §6.2.8.1-3). It should be noticed that this policy essentially applies to the execution of bytecode. Native methods, the Java Card RE itself and possibly some API methods can be granted specific rights or limitations through the FDP_IFF.1.3/JCVM to FDP_IFF.1.5/JCVM elements. The way the Java Card virtual machine manages the transfer of values on the stack and local variables (returned values, uncaught exceptions) from and to internal registers is implementation-dependent. For instance, a returned reference, depending on the implementation of the stack frame, may transit through an internal register prior to being pushed on the stack of the invoker. The returned bytecode would cause more than one OP.PUT

operation under this scheme.

fdp_rip.1/OBJECTS - Subset residual information protection

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

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

fmt_msa.1/JCRE - Management of security attributes

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

*Application Note***:**
The modification of the Selected Applet Context should be performed in accordance with the rules given in JCRE3, §4 and JCVM3, §3.4.

fmt_msa.1/JCVM - Management of security attributes

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

*Application Note***:**
The modification of the Currently Active Context should be performed in accordance with the rules given in JCRE3, §4 and JCVM3, §3.4.

fmt_msa.2/FIREWALL_JCVM - Secure security attributes

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

*Application Note***:**
The following rules are given as examples only. For instance, 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.

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

fmt_msa.3/FIREWALL - Static attribute initialisation

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

fmt_msa.3.2/FIREWALL: The TSF shall not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

Application Note:
FMT_MSA.3.1/FIREWALL

- Objects' security attributes of the access control policy are created and initialized at the creation of the object or the subject. Afterwards, these attributes are no longer mutable (FMT_MSA.1/JCRE). At the creation of an object (OP.CREATE), the newly created object, assuming that the FIREWALL access control SFP permits the operation, gets its Lifetime and Sharing attributes from the parameters of the operation; on the contrary, its Context attribute has a default value, which is its creator's Context attribute and AID respectively (JCRE3, §6.1.3). There is one default value for the Selected Applet Context that is the default applet identifier's Context, and one default value for the Currently Active Context that is "Java Card RE".
- The knowledge of which reference corresponds to a temporary entry point object or a global array and which does not is solely available to the Java Card RE (and the Java Card virtual machine).

FMT_MSA.3.2/FIREWALL

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

fmt_msa.3/JCVM - Static attribute initialisation

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

fmt_msa.3.2/JCVM: The TSF shall not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

fmt_smf.1 - Specification of Management Functions

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

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

fmt_smr.1 - Security roles

fmt_smr.1.1: The TSF shall maintain the roles

- **Java Card RE (JCRE),**
- **Java Card VM (JCVM).**

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

Application Programming Interface

The following SFRs are related to the Java Card API.

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

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

fcs_ckm.1/RSA - Cryptographic key generation

fcs_ckm.1.1/RSA: The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **RSA Key Generation** and specified cryptographic key sizes: **between 512 and the maximum length supported by the card (2048 bits) and multiple of 32 bits** that meet the following: **JCAPI3 extended to support the key lengths specified in VGP**.

fcs_ckm.1/EC - Cryptographic key generation

fcs_ckm.1.1/EC: The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **EC Key Generation (for EC operations over large prime fields - GF(p))** and specified cryptographic key sizes **between 160 and 521 bits** that meet the following: **JCAPI3**.

fcs_ckm.1/DSA - Cryptographic key generation

fcs_ckm.1.1/DSA: The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **DSA Key Generation** and specified cryptographic key sizes **512 bits, 768 bits, 1024 bits** that meet the following: **JCAPI3**.

fcs_ckm.2/DES - Cryptographic key distribution

fcs_ckm.2.1/DES: The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **setKey() of the interface javacard.security.DESKey** that meets the following: **no standard** .

fcs_ckm.2/AES - Cryptographic key distribution

fcs_ckm.2.1/AES: The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **setKey() of the interface javacard.security.AESKey** that meets the following: **no standard** .

fcs_ckm.2/RSA_STD - Cryptographic key distribution

fcs_ckm.2.1/RSA_STD: The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **setExponent() and setModulus() of the interfaces javacard.security.RSAPublicKey and javacard.security.RSAPrivateKey** that meets the following **no standard** .

fcs_ckm.2/RSA_CRT - Cryptographic key distribution

fcs_ckm.2.1/RSA_CRT: The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **setExponent() and setModulus() of the interface javacard.security.RSAPublicKey, and setP(), setQ(), setPQ(), setDP1() , setDQ1() of the interface javacard.security.RSAPrivateCrtKey** that meets the following: **no standard** .

fcs_ckm.2/EC - Cryptographic key distribution

fcs_ckm.2.1/EC: The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **setA(), setB(), setFieldFP, setG(), setK(), setR() of the interface javacard.security.ECKey, setS() of the interface javacard.security.ECPrivateKey, setW of the javacard.security.interface ECPublicKey** that meets the following: **no standard**.

fcs_ckm.2/DSA - Cryptographic key distribution

fcs_ckm.2.1/DSA: The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **setG(), setP(), setQ() of the interface javacard.security.DSAKey, setX() of the interface javacard.security.DSAPrivateKey, setY of the javacard.security.interface DSAPublicKey** that meets the following: **no standard**.

fcs_ckm.3/DES - Cryptographic key access

fcs_ckm.3.1/DES: The TSF shall perform **access to DES Key** in accordance with a specified cryptographic key access method **getKey() of the interface javacard.security.DESKey** that meets the following: **no standard** .

fcs_ckm.3/AES - Cryptographic key access

fcs_ckm.3.1/AES: The TSF shall perform **access to AES Key** in accordance with a specified cryptographic key access method **getKey() of the interface javacard.security.AESKey** that meets the following: **no standard** .

fcs_ckm.3/RSA_STD - Cryptographic key access

fcs_ckm.3.1/RSA_STD: The TSF shall perform **access to RSA Key** in accordance with a specified cryptographic key access method **getExponent() and getModulus() of the interfaces javacard.security.RSAPrivateKey and javacard.security.RSAPublicKey** that meets the following: **no standard** .

fcs_ckm.3/RSA_CRT - Cryptographic key access

fcs_ckm.3.1/RSA_CRT: The TSF shall perform **access to RSA CRT Key** in accordance with a specified cryptographic key access method **getExponent() and getModulus() of the interface javacard.security.RSAPublicKey and getP(),getQ(),getPQ(),getDP1(),getDQ1() of the interfaces javacard.security.RSAPrivateCrtKey** that meets the following: **no standard** **.**

fcs_ckm.3/EC - Cryptographic key access

fcs_ckm.3.1/EC: The TSF shall perform **access to EC FP Key** in accordance with a specified cryptographic key access method **getA(), getB(), getField(), getG(), getK(), getR() of the interface javacard.security.ECKey, getS() of the interface javacard.security.ECPrivateKey, getW() of the javacard.security.interface ECPublicKey** that meets the following: **no standard**.

fcs_ckm.3/DSA - Cryptographic key access

fcs_ckm.3.1/DSA: The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **getG(), getP(), getQ() of the interface javacard.security.DSAKey, getX() of the interface javacard.security.DSAPrivateKey, getY of the javacard.security.interface DSAPublicKey** that meets the following: **no standard**.

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 **clearKey() of the interface javacard.security.Key** that meets the following: **no standard** .

fcs_cop.1/DES-TDES_Cipher - Cryptographic operation

fcs_cop.1.1/DES-TDES_Cipher: The TSF shall perform **data encryption and decryption** in accordance with a specified cryptographic algorithm **Data Encryption Standard (DES, TripleDES) in ECB/CBC Mode** and cryptographic key sizes **64 bits, 128 bits and 192 bits** that meet the following: **FIPS_46-3** .

fcs_cop.1/DES_MAC - Cryptographic operation

fcs_cop.1.1/DES_MAC: The TSF shall perform **4 byte and 8 byte MAC generation and verification** in accordance with a specified cryptographic algorithm **DES in CBC Mode or Triple-DES in outer CBC Mode or Retail MAC** and cryptographic key sizes **64 bits, 128 bits and 192 bits** that meet the following: **ISO_9797-1** .

fcs_cop.1/AES_Cipher - Cryptographic operation

fcs_cop.1.1/AES_Cipher: The TSF shall perform **data encryption and decryption** in accordance with a specified cryptographic algorithm **AES in ECB/CBC Mode** and cryptographic key sizes **128 bits, 192 bits, 256 bits** that meet the following: **FIPS_197** .

fcs_cop.1/AES_MAC - Cryptographic operation

fcs_cop.1.1/AES_MAC: The TSF shall perform **16 byte MAC generation and verification** in accordance with a specified cryptographic algorithm **AES in CBC mode** and cryptographic key sizes **128 bits, 192 bits, 256 bits** that meet the following: **ISO_9797-1**.

fcs_cop.1/RSA_Cipher - Cryptographic operation

fcs_cop.1.1/RSA_Cipher: The TSF shall perform **data encryption and decryption** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **between 512 bits and 2048 bits** that meet the following: **RSA-PKCS1**.

fcs_cop.1/RSA_Signature - Cryptographic operation

fcs_cop.1.1/RSA_Signature: The TSF shall perform **digital signature generation and verification** in accordance with a specified cryptographic algorithm **RSA with SHA** and cryptographic key sizes **between 512 bits and 2048 bits** that meet the following: **RSA-PKCS1**; **ISO_9796-2**; **RSA_PSS.**

fcs_cop.1/EC Signature - Cryptographic operation

fcs_cop.1.1/EC_Signature: The TSF shall perform **digital signature generation and verification** in accordance with a specified cryptographic algorithm **EC with SHA1, SHA224, SHA256, SHA384, SHA512** and cryptographic key sizes **between 160 and 521 bits** that meet the following: **ANSI_X9.62**.

fcs_cop.1/SHA - Cryptographic operation

fcs_cop.1.1/SHA: The TSF shall perform **secure hash computation** in accordance with a specified cryptographic algorithm **SHA1, SHA224, SHA256, SHA384, SHA512** and cryptographic key sizes **none** that meet the following: **FIPS_180-2**.

fcs_cop.1/ECDH_KeyExchange - Cryptographic operation

fcs_cop.1.1/ECDH_KeyExchange: The TSF shall perform **Shared Secret generation and exchange** in accordance with a specified cryptographic algorithm **Elliptic Curve Diffie Hellman** and cryptographic key sizes **between 160 and 521 bits** that meet the following:

IEEEP1363.

fdp_rip.1/ABORT - Subset residual information protection

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

Application Note:
The events that provoke the de-allocation of the previously mentioned references are described in JCRE3, §7.6.3.

fdp_rip.1/APDU - Subset residual information protection

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

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

fdp_rip.1/bArray - Subset residual information protection

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

Application Note:
A resource is allocated to the bArray object when a call to an applet's install() method is performed. There is no conflict with *FDP_ROL.1/FIREWALL* 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.

Editorial Note:
In the second bullet of Application Note, the original reference as stated in the PP (FDP_ROL.1) has been changed in FDP_ROL.1/FIREWALL (in order to be compliant with actual name of the component in the PP and in order to distinguish it from additional requirement fdp_rol.1/Atomicity).

fdp_rip.1/KEYS - Subset residual information protection

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

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

fdp_rip.1/TRANSIENT - Subset residual information protection

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

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

fdp_rol.1/FIREWALL - Basic rollback

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

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

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

Card Security Management

fau_arp.1/JCS - Security alarms

fau_arp.1.1/JCS: The TSF shall take **one of the following actions:**
- **throw an exception**
- **lock the card session**
- **reinitialize the Java Card System and its data**
- **no other actions**

upon detection of a potential security violation.

Refinement:
The "potential security violation" stands for one of the following events:
- CAP file inconsistency,
- typing error in the operands of a bytecode,
- applet life cycle inconsistency,
- card tearing (unexpected removal of the Card out of the CAD) and power failure,
- abort of a transaction in an unexpected context, (see abortTransaction(), [JCAPI3] and [JCRE3], §7.6.2),
- violation of the Firewall or JCVM SFPs,
- unavailability of resources,
- array overflow,
- **other runtime errors related to applet failures (e.g. uncaught exceptions)**

*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.
- Depending on the context of use and the required security level, there are cases where the card manager and the TOE must work in cooperation to detect and appropriately react in case of potential security violation. This behavior must be described in this component. It shall detail the nature of the feedback information provided to the card manager (like the identity of the offending application) and the conditions under which the feedback will occur (any occurrence of the java.lang.SecurityException exception).
- The "locking of the card session" may not appear in the policy of the card manager. Such measure should only be taken in case of severe violation detection; the same holds for the re-initialization of the Java Card System. Moreover, the locking should occur when "clean" re-initialization seems to be impossible.
- The locking may be implemented at the level of the Java Card System as a denial of service (through some systematic "fatal error" message or return value) that lasts up to the next "RESET" event, without affecting other components of the card (such as the card manager). Finally, because the installation of applets is a sensitive process, security alerts in this case should also be carefully considered herein.

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.PIN value, try counter and associated flags,**
- **D.APP_KEYs value and initialized flag,**
- **Elements contained in Secure Arrays (objects instances of class SecureArray)**
- **D.GP_SENSITIVE_DATA**

fdp_sdi.2.2: Upon detection of a data integrity error, the TSF shall **maintain a secure state, deny the use of the corrupted data and/or return an error message**.

Application Note**:**

- Although no such requirement is mandatory in the Java Card specification, at least an exception shall be raised upon integrity errors detection on cryptographic keys, PIN values and their associated security attributes. Even if all the objects cannot be monitored, cryptographic keys and PIN objects shall be considered with particular attention by ST authors as they play a key role in the overall security.

fpr_uno.1/PIN - Unobservability

fpr_uno.1.1/PIN: The TSF shall ensure that **all users and subjects** are unable to observe the operation **all comparison operations** on **D.PIN** by **all users and subjects**.

fpr_uno.1/KEY - Unobservability

fpr_uno.1.1/KEY: The TSF shall ensure that **all users and subjects** are unable to observe the operation **all cryptographic operations** on **D.APP_KEYs** by **all users and subjects**.

fpt_fls.1 - Failure with preservation of secure state

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

Application Note:

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

fpt_tdc.1 - Inter-TSF basic TSF data consistency

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

fpt_tdc.1.2: The TSF shall use

- **the rules defined in [JCVM3] specification**
- **the API tokens defined in the export files of reference implementation,**
- ***The ISO 7816-6 rules***
- ***The rules defined in [*GP211*] Specification***

when interpreting the TSF data from another trusted IT product.

Application Note:

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

AID Management

fia_atd.1/AID - User attribute definition

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

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

Refinement:
"Individual users" stand for applets.

fia_uid.2/AID - User identification before any action

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

Application Note:

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

fia_usb.1/AID - User-subject binding

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

fia_usb.1.2/AID: The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **rules defined in FDP_ACF.1.1/FIREWALL, FMT_MSA.2.1/JCRE, and FMT_MSA.3.1/FIREWALL** .

fia_usb.1.3/AID: The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: **rules defined in FMT_MSA.1.1/JCRE** .
Application Note:
The user is the applet and the subject is the S.PACKAGE. The subject security attribute "Context" shall hold the user security attribute "package AID".

fmt_mtd.1/JCRE - Management of TSF data

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

Application Note:
The Java Card RE manages 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.

fmt_mtd.3/JCRE - Secure TSF data

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

4.6.1.2 ODELG Security Functional Requirements

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

fdp_rip.1/ODEL - Subset residual information protection

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

<u>Application Note:</u>

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

<u>Editorial Note:</u>
In the second bullet of Application Note, the original reference as stated in the PP (FDP_ROL.1) has been changed in FDP_ROL.1/FIREWALL (in order to be compliant with actual name of the component in the PP and in order to distinguish it from additional requirement fdp_rol.1/Atomicity).

fpt_fls.1/ODEL - Failure with preservation of secure state

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

<u>Application Note:</u>
The TOE may provide additional feedback information to the card manager in case of potential security violation (see FAU_ARP.1).

4.6.1.3 GP_API Security Functional Requirements

This group contains the security requirements for the Global Platform API, partially taken form GlobalPlatform Card Security Requirements Specification 1.0.

fdp_acc.1/GP_API - Subset access control

fdp_acc.1.1/ GP_API: The TSF shall enforce the **Global Platform API access control SFP** on **S.APPLICATION**, **O.GP_REGISTRY** and all operations among subjects and objects covered by the SFP. The operations involved in the policy are:

- OP.MANAGE_APP_LIFE_CYCLE_STATE
- OP.GET_CARD_LIFE_CYCLE_STATE
- OP.CARD_LOCK
- OP.CARD_TERMINATE

fdp_acf.1/GP_API - Security attribute based access control

fdp_acf.1.1/GP_API: The TSF shall enforce the **GP API access control SFP** to objects based on the following: **subjects, objects and their security attributes** described hereafter:

Subject/Object	Attributes

S.APPLICATION	AID
O.GP_REGISTRY	ApplicationPrivilege, CardLifeCycleState, ApplicationLifeCycleState, ApplicationAID

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

i. The following operations: **OP.MANAGE_APP_LIFE_CYCLE_STATE, OP.GET_CARD_LIFE_CYCLE_STATE, OP.CARD_LOCK and OP.CARD_TERMINATE** are not allowed to **S.APPLICATION** when **O.GP_REGISTRY[ApplicationLifeCycleState]** is **LOCKED**;

ii. **OP.MANAGE_APP_LIFE_CYCLE_STATE** is only allowed to **S.APPLICATION** if its **AID** is equal to **O.GP_REGISTRY[ApplicationAID]** of the requested Application Life Cycle State**;**

iii. **OP.CARD_LOCK** is only allowed to **S.APPLICATION** if **O.GP_REGISTRY[ApplicationPrivilege]** includes the Card Lock Privilege;

iv. **OP.CARD_TERMINATE** is only allowed to **S.APPLICATION** if **O.GP_REGISTRY[ApplicationPrivilege]** includes the Card Terminate Privilege;

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

fdp_acf.1.4/ GP_API: The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **no rule**.

fmt_msa.1/GP_API - Management of security attributes

fmt_msa.1.1/GP_API: The TSF shall enforce the **GP API access control SFP** to restrict the ability to **modify** the security attributes

- **S.APPLICATION[AID], O.GP_REGISTRY[ApplicationPrivilege]** to **none**
- **O.GP_REGISTRY[ApplicationLifeCycleState], O.GP_REGISTRY[CardLifeCycleState]** to **S.APPLICATION**

fmt_msa.3/GP_API - Static attribute initialization

fmt_msa.3.1/GP_API: The TSF shall enforce the **GP API access control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

fmt_msa.3.2/GP_API: The TSF shall allow the **no roles** to specify alternative initial values to override the default values when an object or information is created.

fmt_smr.1/GP_API - Security roles

fmt_smr.1.1/GP_API: The TSF shall maintain the roles: **Applications**.

fmt_smr.1.2/GP_API: The TSF shall be able to associate users with roles.

fia_uid.1/GP_API - Timing of identification

fia_uid.1.1/GP_API: The TSF shall allow **none** on behalf of the user to be performed before the user is identified.

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

4.6.1.4 SCP Platform Security Functional Requirements

Since the Smart Card Platform belongs to the TOE the following functional requirements (partially taken over from the ST of the certified hardware platform ST23 SB23YR80B which is conformant to PP_0035 and partially taken from [PP_ESforSSD]) are functional requirements for the TOE.

SCP support and recovery

fdp_acc.1/Atomicity - Subset access control

fdp_acc.1.1: The TSF shall enforce the **Access Control Policy for Atomicity** on **Single or multiple WRITE access by S.JCVM and S.JCRE to elements or fields of persistent O.JAVAOBJECT**.

Application Note:
It should be noticed that accessing array's components of a static array, and more generally fields and methods of static objects, is an access to the corresponding O.JAVAOBJECT. Moreover, the conditions under which the operations can be rolled back are stated in FDP_ROL.1/Atomicity.

fdp_rol.1/Atomicity - Basic rollback

fdp_rol.1.1: The TSF shall enforce **Access Control Policy for Atomicity** to permit the rollback of the **Single or multiple WRITE operations** on the **elements or fields of persistent O.JAVAOBJECT**.

fdp_rol.1.2: The TSF shall permit operations to be rolled back within the **bounds of the transaction buffer and, in case of multiple WRITE operations, in the scope between a call to JCSystem.beginTransaction() and one of the following possible events: a call to JCSystem.abortTransaction(), unexpected end of JCVM execution, power loss.**

Application Note:
It should be noticed that accessing array's components of a static array, and more generally fields and methods of static objects, is an access to the corresponding O.JAVAOBJECT. Moreover, the size of transaction buffer is configurable by the Card Administrator during TOE initialization based on customer requests.

fpt_fls.1/Operate - Failure with preservation of secure state

fpt_fls.1.1/Operate: The TSF shall preserve a secure state when the following types of failures occur:
- **loss of power or card tearing,**
- **failed checksum verification on sensitive data,**
- **card life cycle state integrity violation,**
- **EEPROM failure audited through exceptions in the read/write operations and consistency/integrity checks,**
- **potential security violations and abnormal operating conditions detected and made available by the IC and crypto-library.**

IC Hardware

fru_flt.2 - Limited fault tolerance

fru_flt.2.1: The TSF shall ensure the operation of all the TOE's capabilities when the following failures occur: **exposure to operating conditions which are not detected according to the requirement Failure with preservation of secure state (*FPT_FLS.1/SCP*)** .

Editorial Note:
Original reference to FPT_FLS.1 has been changed in FPT_FLS.1/SCP to make it consistent with the renaming of FPT_FLS.1/SCP

fpt_fls.1/SCP - Failure with preservation of secure state

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** .

Refinement:
The term “failure” above also covers “circumstances”. The TOE prevents failures for the “circumstances” defined above. Regarding application note 15 of [PP_0035], the TOE provides information on the operating conditions monitored during Security IC Embedded Software execution and after a warm reset. No audit requirement is however selected in this Security Target.

Editorial Note:
This SFR, drawn from [STlite_SB23] and identified as FPT_FLS.1 has been renamed in FPT_FLS.1/SCP to distinguish it from other instantiation the same SFR (FPT_FLS.1 defined in [PP_JC_Closed], and FPT_FLS.1/Operate defined in this document).

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 **Limited capability and availability policy** .

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 **Limited capability and availability policy** .
Limited capability and availability policy: Deploying Test Features after TOE Delivery does not allow User Data to be disclosed or manipulated, TSF data to be disclosed or manipulated, software to be reconstructed and no substantial information about construction of TSF to be gathered which may enable other attacks.

fpt_php.3 - Resistance to physical attack

fpt_php.3.1: The TSF shall resist **physical manipulation and physical probing** to the **TSF** by responding automatically such that the SFRs are always enforced.

Refinement:
The TSF will implement appropriate mechanisms to continuously counter physical manipulation and physical probing. Due to the nature of these attacks (especially manipulation) the TSF can by no means detect attacks on all of its elements. Therefore, permanent protection against these attacks is required ensuring that security functional requirements are enforced. Hence, “automatic response” means here (i)assuming that there might be an attack at any time and (ii)countermeasures are provided at any time.

fdp_itt.1 - Basic internal transfer protection

fdp_itt.1.1: The TSF shall enforce the **Data Processing Policy** to prevent the **disclosure** of user data when it is transmitted between physically-separated parts of the TOE.

fpt_itt.1 - Basic internal TSF data transfer protection

fpt_itt.1.1: The TSF shall protect TSF data from **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. This requirement is equivalent to FDP_ITT.1 above but refers to TSF data instead of User Data. Therefore, it should be understood as to refer to the same **Data Processing Policy** defined under FDP_IFC.1 below.

fdp_ifc.1 - Subset information flow control

fdp_ifc.1.1: The TSF shall enforce the **Data Processing Policy** on **all confidential data when they are processed or transferred by the TSF or by the Security IC Embedded Software**.

fcs_rng.1 - Generation of random numbers

fcs_rng.1.1: The TSF shall provide a **physical** random number generator that implements: **total failure test of the random source** .

fcs_rng.1.2: The TSF shall provide random numbers that meet **P2 class of BSI_AIS31** .

4.6.1.5 Additional Security Functional Requirements

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 and periodically during normal operation** to demonstrate the correct operation of **the TSF**.

fpt_tst.1.2: The TSF shall provide authorised users with the capability to verify the integrity of **TSF data**.

fpt_tst.1.3: The TSF shall provide authorised users with the capability to verify the integrity of **parts of TSF (TSF executable code)**.

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.CRYPTO** and **User Data: D_APP_KEYs, D.PIN**.

fpt_emsec.1.2: The TSF shall ensure **unauthorized users** are unable to use the following interface **electrical contacts** to gain access to **TSF Data: D.CRYPTO** and **User Data: D_APP_KEYs, D.PIN**.

4.6.2 TOE Security assurance requirements

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_IMP.1 - Implementation representation of the TSF

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 packages 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 applets (through shareable interface mechanism, for instance) must include some kind of authentication of the involved parties, even when no sensitive information seems at stake (so-called “defensive development”).

AGD_PRE.1 - Preparative procedures
This SAR ensures the integrity of the TOE and its documentation during the transfer of the TOE between all the actors appearing in the first two stages. Procedures shall ensure protection of TOE material/information under delivery and storage that corrective actions are taken in case of improper operation in the delivery process and storage and that people dealing with the procedure for delivery have the required skills.

ALC_CMC.4 - Production support, acceptance procedures and automation

ALC_CMS.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.

4.6.3 Security requirements rationale

4.6.3.1 OBJECTIVES

Identification

O.SID
Subjects' identity is AID-based (applets, packages), and is met by the following SFRs: FIA_ATD.1/AID, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MTD.1/JCRE and FMT_MTD.3/JCRE.
Lastly, installation procedures ensure protection against forgery (the AID of an applet is under the control of the TSFs) or re-use of identities (FIA_UID.2/AID, FIA_USB.1/AID).

O.ROLES
Users' identification and association with corresponding roles is ensured by FMT_SMR.1/GP_API

Execution

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

O.GLOBAL_ARRAYS_CONFID
Only arrays can be designated as global, and the only global arrays required in the Java Card API are the APDU buffer and the byte array input parameter (bArray) to an applet's install method. The clearing requirement of those arrays is met by FDP_RIP.1/APDU and FDP_RIP.1/bArray respectively. The JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) prevents an application from keeping a pointer to a shared buffer, which could be used to read its contents when the buffer is being used by another application.
If the TOE provides JCRMI functionality, protection of the array parameters of remotely invoked methods, which are global as well, is covered by the general initialization of method parameters (FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS and FDP_RIP.1/TRANSIENT).

O.GLOBAL_ARRAYS_INTEG
This objective is met by the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM), which prevents an application from keeping a pointer to the input/output buffer of the card, or any other global array that is shared by all the applications. Such a pointer could be used to access and modify it when the buffer is being used by another application.

O.NATIVE
This security objective is covered by FDP_ACF.1/FIREWALL: the only means to execute native code is the invocation of a Java Card API method.

O.OPERATE
The TOE is protected in various ways against applets' actions (FPT_TDC.1), the FIREWALL access control policy (FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL), and is able to detect and block various failures or security violations during usual working (FPT_FLS.1/JCS, FPT_FLS.1/ODEL, FAU_ARP.1/JCS). Its security-critical parts and procedures are also protected: applets' installation may be cleanly aborted (FDP_ROL.1/FIREWALL), communication with external users and their internal subjects is well-controlled (FIA_ATD.1/AID and FIA_USB.1/AID) to prevent alteration of TSF data (also protected by components of the FPT class).
Almost every objective and/or functional requirement indirectly contributes to this one too.

<u>Application Note</u>**: In addition, protection of the start-up phase of the TOE (TSF-testing) is also indirectly covered by FPT_TST.1. Functional requirements that indirectly contribute to satisfy this objective are: FDP_ACC.1/ATOMICITY, FDP_ROL.1/ATOMICITY, FPT_FLS.1/OPERATE, FPT_FLS.1/SCP.**

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

O.RESOURCES
The TSFs detects stack/memory overflows during execution of applications (FAU_ARP.1/JCS, FPT_FLS.1/JCS, FPT_FLS.1/ODEL). Failed installations are not to create memory leaks

(FDP_ROL.1/FIREWALL) as well. Memory management is controlled by the TSF (FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_SMR.1 and FMT_SMF.1).

O.SCP.RECOVERY
In case of loss of power, or if the smart card is withdrawn from the CAD while an operation is in progress, mechanisms implemented in the TSF allow to recover to a consistent and secure state (FPT_FLS.1/OPERATE), eventually completing the interrupted operation or rolling it back (FDP_ROL.1/ATOMICITY).

O.SCP.IC
This objective is automatically satisfied by security requirements drawn from the Security Target Lite of the certified secure microcontroller ([STlite_SB23]) and listed in section 4.6.1.4 – IC Hardware (i.e: FPT_FLS.1, FRU_FLT.2, FPT_FLS.1/SCP, FMT_LIM.1, FMT_LIM.2, FPT_PHP.3, FDP_ITT.1, FPT_ITT.1, FDP_IFC.1).

O.SCP.SUPPORT
TSF Secure mechanism for storing data in EEPROM memory and supporting transactional updates has been defined by FDP_ACC.1/ATOMICITY and FDP_ROL.1/ATOMICITY. The SCP portion of the TSF, relying on the certified secure microcontroller and associated crypto-library ([STlite_SB23]), provides low-level cryptographic primitives to allow implementation of the Java Card System cryptography services (see O.CIPHER). In addition the SCP provides self-test mechanisms (as defined by FPT_TST.1) and preserves a secure state in case of security violations detected by low-level routines or by the IC (FPT_FLS.1/Operate).

Services

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

O.CIPHER
This security objective is directly covered by the instantiations of FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4 and FCS_COP.1. The SFR FPR_UNO.1 contributes in covering this security objective and controls the observation of the cryptographic operations which may be used to disclose the keys.
The FCS_RNG.1 also contributes in covering O.CIPHER because random generation is provided as a cryptographic service to applications and plays an important role in cryptographic algorithms and countermeasures.

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

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

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

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

Embedded Software

O.SIDE_CHANNEL
This security objective requires the TOE to protect confidential user and TSF data stored and/or processed in the smart card IC, against disclosure by measurement and analysis of the shape and amplitude of signals or the time between events found by measuring signals on the electromagnetic field, power consumption, clock, or I/O lines, which is addressed by the SFR FPT_EMSEC.1.

O.LIFE_CYCLE
This is ensured by the policy defined in FDP_ACC.1/GP_API and FDP_ACF.1/GP_API, the TOE manages its own life cycle states as well as transitions between them and rejects operations unexpected in its current life cycle.

4.6.3.2 Rationale tables of security objectives and SFRs

Component	Objectives
fdp_acc.2/FIREWALL	O.FIREWALL O.OPERATE O.PIN-MNGT
fdp_acf.1/FIREWALL	O.FIREWALL O.NATIVE O.OPERATE O.PIN-MNGT
fdp_ifc.1/JCVM	O.FIREWALL O.GLOBAL_ARRAYS_CONFID O.GLOBAL_ARRAYS_INTEG
fdp_iff.1/JCVM	O.FIREWALL O.GLOBAL_ARRAYS_CONFID O.GLOBAL_ARRAYS_INTEG
fdp_rip.1/OBJECTS	O.GLOBAL_ARRAYS_CONFID O.REALLOCATION O.KEY-MNGT O.PIN-MNGT O.TRANSACTION
fmt_msa.1/JCRE	O.SID O.FIREWALL
fmt_msa.1/JCVM	O.SID O.FIREWALL
fmt_msa.2/FIREWALL_JCVM	O.FIREWALL
fmt_msa.3/FIREWALL	O.SID O.FIREWALL
fmt_msa.3/JCVM	O.SID O.FIREWALL

fmt_smf.1	O.FIREWALL O.RESOURCES
fmt_smr.1	O.FIREWALL O.RESOURCES
fcs_ckm.1/RSA	O.CIPHER O.KEY-MNGT
fcs_ckm.1/EC	O.CIPHER O.KEY-MNGT
fcs_ckm.1/DSA	O.CIPHER O.KEY-MNGT
fcs_ckm.2/DES	O.CIPHER O.KEY-MNGT
fcs_ckm.2/AES	O.CIPHER O.KEY-MNGT
fcs_ckm.2/RSA_STD	O.CIPHER O.KEY-MNGT
fcs_ckm.2/RSA_CRT	O.CIPHER O.KEY-MNGT
fcs_ckm.2/EC	O.CIPHER O.KEY-MNGT
fcs_ckm.2/DSA	O.CIPHER O.KEY-MNGT
fcs_ckm.3/DES	O.CIPHER O.KEY-MNGT
fcs_ckm.3/AES	O.CIPHER O.KEY-MNGT
fcs_ckm.3/RSA_STD	O.CIPHER O.KEY-MNGT
fcs_ckm.3/RSA_CRT	O.CIPHER O.KEY-MNGT
fcs_ckm.3/EC	O.CIPHER O.KEY-MNGT
fcs_ckm.3/DSA	O.CIPHER O.KEY-MNGT
fcs_ckm.4	O.CIPHER O.KEY-MNGT
fcs_cop.1/DES-TDES_Cipher	O.CIPHER O.KEY-MNGT
fcs_cop.1/DES_MAC	O.CIPHER O.KEY-MNGT
fcs_cop.1/AES_Cipher	O.CIPHER O.KEY-MNGT
fcs_cop.1/AES_MAC	O.CIPHER O.KEY-MNGT
fcs_cop.1/RSA_Cipher	O.CIPHER O.KEY-MNGT
fcs_cop.1/RSA_Signature	O.CIPHER O.KEY-MNGT
fcs_cop.1/EC_Signature	O.CIPHER O.KEY-MNGT
fcs_cop.1/SHA	O.CIPHER O.KEY-MNGT
fcs_cop.1/ECDH_KeyExchange	O.CIPHER O.KEY-MNGT

fdp_rip.1/ABORT	O.GLOBAL_ARRAYS_CONFID O.REALLOCATION O.KEY-MNGT O.PIN-MNGT O.TRANSACTION
fdp_rip.1/APDU	O.GLOBAL_ARRAYS_CONFID O.REALLOCATION O.KEY-MNGT O.PIN-MNGT O.TRANSACTION
fdp_rip.1/bArray	O.GLOBAL_ARRAYS_CONFID O.REALLOCATION O.KEY-MNGT O.PIN-MNGT O.TRANSACTION
fdp_rip.1/KEYS	O.GLOBAL_ARRAYS_CONFID O.REALLOCATION O.KEY-MNGT O.PIN-MNGT O.TRANSACTION
fdp_rip.1/TRANSIENT	O.GLOBAL_ARRAYS_CONFID O.REALLOCATION O.KEY-MNGT O.PIN-MNGT O.TRANSACTION
fdp_rol.1/FIREWALL	O.OPERATE O.RESOURCES O.PIN-MNGT O.TRANSACTION
fau_arp.1/JCS	O.OPERATE O.RESOURCES O.ALARM
fdp_sdi.2	O.KEY-MNGT O.PIN-MNGT
fpr_uno.1/PIN	O.CIPHER O.KEY-MNGT O.PIN-MNGT
fpr_uno.1/KEY	O.CIPHER O.KEY-MNGT O.PIN-MNGT
fpt_fls.1	O.OPERATE O.RESOURCES O.ALARM
fpt_tdc.1	O.OPERATE
fia_atd.1/AID	O.SID O.OPERATE
fia_uid.2/AID	O.SID
fia_usb.1/AID	O.SID O.OPERATE
fmt_mtd.1/JCRE	O.SID O.FIREWALL O.RESOURCES
fmt_mtd.3/JCRE	O.SID O.FIREWALL O.RESOURCES

fdp_rip.1/ODEL	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.OBJ-DELETION
fpt_fls.1/ODEL	O.OPERATE, O.RESOURCES, O.ALARM, O.OBJ-DELETION
fdp_acc.1/GP_API	O.LIFE_CYCLE
fdp_acf.1/GP_API	O.LIFE_CYCLE
fmt_msa.1/GP_API	O.LIFE_CYCLE
fmt_msa.3/GP_API	O.LIFE_CYCLE
fmt_smr.1/GP_API	O.ROLES
fia_uid.1/GP_API	O.LIFE_CYCLE
fdp_acc.1/Atomicity	O.OPERATE (indirectly) *O.SCP.SUPPORT*
fdp_rol.1/Atomicity	O.OPERATE (indirectly) *O.SCP.RECOVERY* *O.SCP.SUPPORT*
fpt_fls.1/Operate	O.OPERATE (indirectly) *O.SCP.RECOVERY* *O.SCP.SUPPORT*
fru_flt.2	*O.SCP.IC*
fpt_fls.1/SCP	O.OPERATE O.RESOURCES O.ALARM *O.SCP.IC*
fmt_lim.1	*O.SCP.IC*
fmt_lim.2	*O.SCP.IC*
fpt_php.3	*O.SCP.IC*
fdp_itt.1	*O.SCP.IC*
fpt_itt.1	*O.SCP.IC*
fdp_ifc.1	*O.SCP.IC*
fcs_rng.1	O.CIPHER
fpt_tst.1	O.OPERATE (indirectly) *O.SCP.SUPPORT*
fpt_emsec.1	O.SIDE_CHANNEL

Table 13 - Security Functional Requirements (SFR) vs. Objectives

4.6.3.3 Dependencies

<u>**SFRs dependencies**</u>

Requirements	CC Dependencies	Satisfied Dependencies
fdp_acc.2/FIREWALL	fdp_acf.1	fdp_acf.1/FIREWALL
fdp_acf.1/FIREWALL	fdp_acc.1 fmt_msa.3	fdp_acc.2/FIREWALL fmt_msa.3/FIREWALL
fdp_ifc.1/JCVM	fdp_iff.1	fdp_iff.1/JCVM
fdp_iff.1/JCVM	fdp_ifc.1 fmt_msa.3	fdp_ifc.1/JCVM fmt_msa.3/JCVM

fdp_rip.1/OBJECTS	No dependencies	No dependencies
fmt_msa.1/JCRE	fmt_smr.1 fmt_smf.1 One of : fdp_acc.1, fdp_ifc.1	fdp_acc.2/FIREWALL fmt_smr.1 fmt_smf.1 dependency is not supported (see below)
fmt_msa.1/JCVM	fmt_smr.1 fmt_smf.1 One of : fdp_acc.1, fdp_ifc.1	fmt_smr.1 fmt_smf.1 fdp_acc.2/FIREWALL fdp_ifc.1/JCVM
fmt_msa.2/FIREWALL_JCVM	fmt_msa.1 fmt_smr.1 One of : fdp_acc.1 fdp_ifc.1	fmt_msa.1/JCVM fmt_msa.1/JCRE fmt_smr.1 fdp_acc.2/FIREWALL fdp_ifc.1/JCVM
fmt_msa.3/FIREWALL	fmt_msa.1 fmt_smr.1	fmt_msa.1/JCRE fmt_msa.1/JCVM fmt_smr.1
fmt_msa.3/JCVM	fmt_msa.1 fmt_smr.1	fmt_msa.1/JCVM fmt_smr.1
fmt_smf.1	No dependencies	No dependencies
fmt_smr.1	fia_uid.1	fia_uid.2/AID
fcs_ckm.1/RSA fcs_ckm.1/EC fcs_ckm.1/DSA	fcs_ckm.4 One of : fcs_ckm.2 fcs_cop.1	fcs_ckm.4 fcs_ckm.2
fcs_ckm.2/DES fcs_ckm.2/AES fcs_ckm.2/RSA_STD fcs_ckm.2/RSA_CRT fcs_ckm.2/EC fcs_ckm.2/DSA	fcs_ckm.4 One of : fdp_itc.1 fdp_itc.2 fcs_ckm.1	fcs_ckm.4 fcs_ckm.1/EC fcs_ckm.1/RSA fcs_ckm.1/DSA
fcs_ckm.3/DES fcs_ckm.3/AES fcs_ckm.3/RSA_STD fcs_ckm.3/RSA_CRT fcs_ckm.3/EC fcs_ckm.3/DSA	fcs_ckm.4 One of : fdp_itc.1 fdp_itc.2 fcs_ckm.1	fcs_ckm.4 fcs_ckm.1/EC fcs_ckm.1/RSA fcs_ckm.1/DSA
fcs_ckm.4	One of : fdp_itc.1 fdp_itc.2 fcs_ckm.1	fcs_ckm.1/EC fcs_ckm.1/RSA fcs_ckm.1/DSA
fcs_cop.1/DES-TDES_Cipher fcs_cop.1/DES_MAC fcs_cop.1/AES_Cipher fcs_cop.1/AES_MAC fcs_cop.1/RSA_Cipher fcs_cop.1/RSA_Signature fcs_cop.1/EC_Signature fcs_cop.1/SHA fcs_cop.1/ECDH_KeyExchange	fcs_ckm.4 One of : fdp_itc.1 fdp_itc.2 fcs_ckm.1	fcs_ckm.4 fcs_ckm.1/EC fcs_ckm.1/RSA fcs_ckm.1/DSA
fdp_rip.1/ABORT	No dependencies	No dependencies
fdp_rip.1/APDU	No dependencies	No dependencies
fdp_rip.1/bArray	No dependencies	No dependencies
fdp_rip.1/KEYS	No dependencies	No dependencies
fdp_rip.1/TRANSIENT	No dependencies	No dependencies
fdp_rol.1/FIREWALL	One of: fdp_acc.1 fdp_ifc.1	fdp_acc.2/FIREWALL, fdp_ifc.1/JCVM
fau_arp.1/JCS	fau_saa.1	Not supported (see below)

fdp_sdi.2	No dependencies	No dependencies
fpr_uno.1/PIN	No dependencies	No dependencies
fpr_uno.1/KEY	No dependencies	No dependencies
fpt_fls.1	No dependencies	No dependencies
fpt_tdc.1	No dependencies	No dependencies
fia_atd.1/AID	No dependencies	No dependencies
fia_uid.2/AID	No dependencies	No dependencies
fia_usb.1/AID	fia_atd.1	fia_atd.1/AID
fmt_mtd.1/JCRE	fmt_smr.1 fmt_smf.1	fmt_smr.1 fmt_smf.1
fmt_mtd.3/JCRE	fmt_mtd.1	fmt_mtd.1/JCRE
fdp_rip.1/ODEL	No dependencies	No dependencies
fpt_fls.1/ODEL	No dependencies	No dependencies
fdp_acc.1/GP_API	fdp_acf.1	fdp_acf.1/GP_API
fdp_acf.1/GP_API	One of : fdp_acc.1 fdp_ifc.1	fdp_acc.1/GP_API
fmt_msa.1/GP_API	fmt_smr.1 fmt_smf.1 One of : fdp_acc.1 fdp_ifc.1	fmt_smr.1/GP_API fmt_smf.1 not supported fdp_acc.1/GP_API
fmt_msa.3/GP_API	fmt_msa.1 fmt_smr.1	fmt_msa.1/GP_API fmt_smr.1/GP_API
fmt_smr.1/GP_API	fia_uid.1	fia_uid.1/GP_API
fia_uid.1/GP_API	No dependencies	No dependencies
fdp_acc.1/Atomicity	fdp_acf.1	Not supported (see below)
fdp_rol.1/Atomicity	One of : fdp_acc.1 fdp_ifc.1	fdp_acc.1/Atomicity
fpt_fls.1/Operate	No dependencies	No dependencies
fru_flt.2	fpt_fls.1/SCP	fpt_fls.1/SCP
fpt_fls.1/SCP	No dependencies	No dependencies
fmt_lim.1	fmt_lim.2	fmt_lim.2
fmt_lim.2	fmt_lim.1	fmt_lim.1
fpt_php.3	No dependencies	No dependencies
fdp_itt.1	One of : fdp_acc.1 fdp_ifc.1	fdp_ifc.1
fpt_itt.1	No dependencies	No dependencies
fdp_ifc.1	fdp_iff.1	Not supported (see below)
fcs_rng.1	No dependencies	No dependencies
fpt_tst.1	No dependencies	No dependencies
fpt_emsec.1	fpt_emsec.1	fpt_emsec.1

Table 14 - Security Functional Requirements (SFRs) dependencies

Rationale for the exclusion of dependencies

The dependency FMT_SMF.1 of FMT_MSA.1/JCRE is unsupported. The dependency between FMT_MSA.1/JCRE and FMT_SMF.1 is not satisfied because no management functions are required for the Java Card RE.
The dependency FAU_SAA.1 of FAU_ARP.1 is unsupported. Potential violation analysis is used to specify the set of auditable events whose occurrence or accumulated occurrence held to indicate a potential violation of the SFRs, and any rules to be used to perform the violation analysis. The dependency of FAU_ARP.1/JCS on this functional requirement assumes that a "potential security violation" is an audit event indicated by the FAU_SAA.1 component. The events listed in FAU_ARP.1/JCS are, on the contrary, merely self-contained ones (arithmetic exception, ill-formed bytecodes, access failure) and ask for a straightforward reaction of the TSFs on their occurrence at runtime. The JCVM or other components of the TOE detect these events during their usual working order. Thus, in principle there would be no applicable audit recording in this framework.
Moreover, no specification of one such recording is provided elsewhere. Therefore no set of auditable events could possibly be defined.

The dependency FMT_SMF.1 of FMT_MSA.1/GP_API is unsupported. The dependency between FMT_MSA.1/GP_API and FMT_SMF.1 is not satisfied because no management functions are required for the GP API.
The dependency FDP_ACF.1 of FDP_ACC.1/Atomicity is unsupported. The FDP_ACC requirement serves as a framework for the definition of the operations that can be rolled back. There is no need to require FDP_ACF since there is neither mandatory attribute nor rule.
The dependency FDP_IFF.1 of FDP_IFC.1 is unsupported (taken from PP_0035). Part 2 of the Common Criteria defines the dependency of FDP_IFC.1 (information flow control policy statement) on FDP_IFF.1 (Simple security attributes). The specification of FDP_IFF.1 would not capture the nature of the security functional requirement 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).

<u>SARs dependencies</u>

EAL5+ Assurance Requirements	CC Dependencies	Satisfied Dependencies
ADV_ARC.1	(ADV_FSP.1) and (ADV_TDS.1)	ADV_FSP.5, ADV_TDS.4
ADV_FSP.5	(ADV_IMP.1) and (ADV_TDS.1)	ADV_IMP.1, ADV_TDS.4
ADV_IMP.1	(ADV_TDS.3) and (ALC_TAT.1)	ADV_TDS.4, ALC_TAT.2
ADV_INT.2	(ADV_IMP.1) and (ADV_TDS.3) and (ALC_TAT.1)	ADV_IMP.1, ADV_TDS.4, ALC_TAT.2
ADV_TDS.4	(ADV_FSP.5)	ADV_FSP.5
AGD_OPE.1	(ADV_FSP.1)	ADV_FSP.5
AGD_PRE.1	No dependencies	
ALC_CMC.4	(ALC_CMS.1) and (ALC_DVS.1) and (ALC_LCD.1)	ALC_CMS.4, ALC_DVS.2,) ALC_LCD.1
ALC_CMS.5	No dependencies	
ALC_DEL.1	No dependencies	
ALC_DVS.2	No dependencies	
ALC_LCD.1	No dependencies	
ALC_TAT.2	(ADV_IMP.1)	ADV_IMP.1
ASE_CCL.1	(ASE_ECD.1) and (ASE_INT.1) and (ASE_REQ.1)	ASE_ECD.1, ASE_INT.1, ASE_REQ.2
ASE_ECD.1	No dependencies	
ASE_INT.1	No dependencies	
ASE_OBJ.2	(ASE_SPD.1)	ASE_SPD.1
ASE_REQ.2	(ASE_ECD.1) and (ASE_OBJ.2)	ASE_ECD.1, ASE_OBJ.2
ASE_SPD.1	No dependencies	
ASE_TSS.1	(ADV_FSP.1) and (ASE_INT.1) and (ASE_REQ.1)	ADV_FSP.5, ASE_INT.1, ASE_REQ.2
ATE_COV.2	(ADV_FSP.2) and (ATE_FUN.1)	ADV_FSP.5, ATE_FUN.1
ATE_DPT.3	(ADV_ARC.1) and (ADV_TDS.4) and (ATE_FUN.1)	ADV_ARC.1, ADV_TDS.4, ATE_FUN.1
ATE_FUN.1	(ATE_COV.1)	ATE_COV.2
ATE_IND.2	(ADV_FSP.2) and (AGD_OPE.1) and (AGD_PRE.1) and (ATE_COV.1) and (ATE_FUN.1)	ADV_FSP.5, AGD_OPE.1, AGD_PRE.1, ATE_COV.2, ATE_FUN.1
AVA_VAN.5	(ADV_ARC.1) and (ADV_FSP.4) and (ADV_IMP.1) and (ADV_TDS.3) and (AGD_OPE.1) and (AGD_PRE.1) and (ATE_DPT.1)	ADV_ARC.1, ADV_FSP.5, ADV_IMP.1, ADV_TDS.4, AGD_OPE.1, AGD_PRE.1, ATE_DPT.3

Table 15 - Security Assurance Requirements (SARs) dependencies

Rationale for the Security Assurance Requirements

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

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 and the embedding product. The standard ALC_DVS.1 requirement mandated by EAL5 is not enough.

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

AVA_VAN.5 Advanced methodical Vulnerability Analysis
The TOE is intended to operate in hostile environments. AVA_VAN.5 "Advanced methodical vulnerability analysis" is considered as the expected level for Java Card technology-based products hosting sensitive applications, in particular in payment and identity areas. AVA_VAN.5 has dependencies on ADV_ARC.1, ADV_FSP.1, ADV_TDS.3, ADV_IMP.1, AGD_PRE.1 and AGD_OPE.1. All of them are satisfied by EAL5.

4.7 TOE summary specification (ASE_TSS)

4.7.1 Statement of the TOE security functionality

4.7.1.1 SF.Firewall

This security functionality enforces the FIREWALL access control SFP and JCVM information flow control SFP.
Its main responsibilities are:

- to check inter-context access;
- to manage access to JCRE Entry Point Object and Global Array so to control the applets access to system resources;
- to control Shareable access so to permit inter-applet resources sharing in a secure and aware way;
- to check temporary references storing: Temporary JCRE Entry Point Object references cannot be stored in any static fields.

4.7.1.2 SF.SecureManagement

Secure Management Security Functionality main responsibilities are:

- Providing the JCRE and JCVM functionalities:
  - APDU handling and forwarding to application,
  - Application selection process,
  - Consistent execution of bytecodes
- Memory cleaning upon: allocation of class instances, arrays, and APDU buffer, and de-allocation of bArray object, any transient object, any reference to an object instance created during an aborted transaction.
- Guaranteeing, with the support of SF.SmartCardPlatform, that operations on secret keys and PIN codes are not observable by other subjects by observation of variations in power consumption or timing analysis.
- Preservation of a secure state and taking appropriate action when the following types of failures occur:

  - loss of power or card tearing,
  - applet and card life cycle inconsistencies,
  - EEPROM failure audited through exceptions in the read/write operations and consistency/integrity checks,
  - potential security violations and abnormal operating conditions detected and made available by the IC and crypto-library,
  - corruption of security attributes of objects and arrays,
  - corruption of value of sensitive data.
- Secure management of security attributes:
  - Selected Applet Context
  - Active Applet,
  - Active Context,
  - List of registered applets’ AIDs,
  - Package AID,
  - Applet’s version number,
  - Applet Selection Status,
  - Application Privileges,
  - Application Life Cycle State,
  - Card Life Cycle State
- Permitting operations roll-back enforcing the FIREWALL access control SFP.

This security functionality also provides service API to applets for secure management of applet life cycle and integrity protected arrays.

4.7.1.3 SF.CryptoKey

This security functionality is related to all the operations on the application keys.
In particular it manages:
- key distribution;
- key access;
- key destruction;
- key generation;

SF.CryptoKey also assures key integrity.
The security functionality relies on a cryptolibrary provided by the hardware manufacturer to perform the RSA and the EC key generation. The cryptolibrary uses an hardware random number generator in order to generate prime numbers and keys (see SF.SmartCardPlatform).
This security functionality is also provided as a service by API to applets.

4.7.1.4 SF.CryptoOp

This security functionality provides to users the cryptographic support to perform encryption/decryption and signature/verification.
The cryptographic functionalities provided are:
- DES-ECB in (encryption/decryption)
- DES-CBC in (encryption/decryption)
- Triple DES-ECB in (encryption/decryption) with 16, 24 bytes of key
- Triple DES-CBC in (encryption/decryption) with 16, 24 bytes of key
- AES-ECB in (encryption/decryption) with 128, 192, 256 bits of key
- AES-CBC in (encryption/decryption) with 128, 192, 256 bits of key
- RSA both with key in standard and CRT mode. Supported key length up to 2048 bits

- EC over GF(p) signature calculation and verification with key length up to 521 bits
- SHA Digest Calculation with digest sizes of 160 (SHA-1), 224, 256, 384, 512 bits

The security functionality also provides means for random number generation, Diffie-Hellman and Elliptic Curve Diffie-Hellman shared secret generation and exchange.

In order to avoid disclosure of secret values, memory areas containing cryptographic keys and data used in the calculation are cleared after the execution of crypto operations.
The security functionality relies on the operating system software and hence on crypto library provided by the hardware manufacturer in order to perform the elementary cryptographic computations (see SF.SmartCardPlatform).
This security functionality is also provided as a service by API to applets.

4.7.1.5 SF.Transaction

Controls all the operations concerning “persistent memory” updates in order to guarantee the coherence of the sensitive data if a failure occurs during the operations.
This SF has two main responsibilities:

- Grant the atomic updates of class fields, instance fields and elements of persistent arrays;
- Provide to the TOE the support for Java Card transactional mechanism;

The mechanism relies on dedicated buffers in EEPROM.
EEPROM data affected by the update operation are either completely updated or reverted at the value they had at the beginning of the operation.
This security functionality is also provided as a service by API to applets.

4.7.1.6 SF.PIN

This security functionality is related to all the operation related to PIN objects.
In particular SF.PIN:

- provides means to perform PIN Verification;
- automatically decreases the try check counter of PINs in case of PIN verification failure;
- provides the functionality to update PIN value and the try counter.

PIN verification procedure consists in the comparison of the PIN provided by the user application requesting the verification procedure with the PIN stored into a PIN object.

This security functionality also guarantees the integrity of the stored PIN value, try counter and verification status.
This security functionality is also offered as a service by API to applets.

4.7.1.7 SF.ObjectDeletion

This security functionality is in charge of de-allocation of memory resources of objects no longer accessible from the installed packages and applet instances. Clearing is only performed when applet invokes JCSystem.requestObjectDeletion() method. The security functionality also guarantees that, once the method has been invoked, information content of unreachable objects cannot be retrieved anymore.
This security functionality is also offered as a service by API to applets.

4.7.1.8 SF.SmartCardPlatform

The certified hardware and associated crypto library feature the following Security Functionalities used by this composite TOE. The exact formulation can be found in the hardware security target [STlite_SB23]:

- Hardware secure initialization and correct reset operation management (TSF_INIT_A).
- TOE logical integrity verifying valid CPU usage, checking memory, code and data integrity, monitoring various manifestations of fault injection attempts (TSF_INT_A).

- Physical tampering protection, ensuring: detection of clock and voltage supply operating changes by the environment, detection of attempts to violate its physical integrity, and glitch attacks (TSF_PHT_A).
- Information leakage protection to guarantee unobservability (TSF_OBS_A).
- Symmetric Key Cryptography Support for AES, DES and Triple DES (TSF_SKCS_A).
- Basic functions for Elliptic Curves Cryptography over prime fields and asymmetric key cryptography algorithms (RSA), through the TSF service TSF_AKCS_A used by SF.CryptoOp and SF.CryptoKey.
- Random Number Generation (TSF_ALEAS_A) used by SF.CryptoKey and SF.CryptoOp.
- Management of security violations attempts (TSF_ADMINIS_A) like incorrect CPU usage, integrity loss in NVM, ROM or RAM, code signature alarm, fault injection attempt, watchdog timeout, access attempt to unavailable or reserved memory areas, MPU errors, clock and voltage supply operating changes by the environment, TOE physical integrity abuse.

4.7.2 TOE summary specification rationale

The following table provides a list of the TOE Security Functionalities (SF) and the coverage of the SFRs by the SFs.

SFR \ SF	SF.Firewall	SF.SecureManagement	SF.CryptoKey	SF.CryptoOp	SF.Transaction	SF.ObjectDeletion	SF.PIN	SF.SmartCardPlatform
fdp_acc.2/FIREWALL	X							
fdp_acf.1/FIREWALL	X							
fdp_ifc.1/JCVM	X							
fdp_iff.1/JCVM	X							
fdp_rip.1/OBJECTS		X						
fmt_msa.1/JCRE		X						
fmt_msa.1/JCVM		X						
fmt_msa.2/FIREWALL_JCVM		X						
fmt_msa.3/FIREWALL		X						
fmt_msa.3/JCVM		X						
fmt_smf.1		X						
fmt_smr.1		X						
fcs_ckm.1/RSA			X					X
fcs_ckm.1/EC			X					X
fcs_ckm.1/DSA			X					X
fcs_ckm.2/DES			X					
fcs_ckm.2/AES			X					
fcs_ckm.2/RSA_STD			X					
fcs_ckm.2/RSA_CRT			X					
fcs_ckm.2/EC			X					
fcs_ckm.2/DSA			X					
fcs_ckm.3/DES			X					
fcs_ckm.3/AES			X					
fcs_ckm.3/RSA_STD			X					
fcs_ckm.3/RSA_CRT			X					
fcs_ckm.3/EC			X					
fcs_ckm.3/DSA			X					

SFR \ SF	SF.Firewall	SF.SecureManagement	SF.CryptoKey	SF.CryptoOp	SF.Transaction	SF.ObjectDeletion	SF.PIN	SF.SmartCardPlatform
fcs_ckm.4			X					
fcs_cop.1/DES-TDES_Cipher				X				X
fcs_cop.1/DES_MAC				X				X
fcs_cop.1/AES_Cipher				X				X
fcs_cop.1/AES_MAC				X				X
fcs_cop.1/RSA_Cipher				X				X
fcs_cop.1/RSA_Signature				X				X
fcs_cop.1/EC_Signature				X				X
fcs_cop.1/SHA				X				X
fcs_cop.1/ECDH_KeyExchange				X				X
fdp_rip.1/ABORT		X						
fdp_rip.1/APDU		X						
fdp_rip.1/bArray		X						
fdp_rip.1/KEYS				X				X
fdp_rip.1/TRANSIENT		X						
fdp_rol.1/FIREWALL		X			X			
fau_arp.1/JCS		X						
fdp_sdi.2		X	X				X	
fpr_uno.1/PIN		X						X
fpr_uno.1/KEY		X						X
fpt_fls.1		X						
fpt_tdc.1		X						
fia_atd.1/AID		X						
fia_uid.2/AID		X						
fia_usb.1/AID		X						
fmt_mtd.1/JCRE		X						
fmt_mtd.3/JCRE		X						
fdp_rip.1/ODEL						X		
fpt_fls.1/ODEL						X		
fdp_acc.1/GP_API		X						
fdp_acf.1/GP_API		X						
fmt_msa.1/GP_API		X						
fmt_msa.3/GP_API		X						
fmt_smr.1/GP_API		X						
fia_uid.1/GP_API		X						
fdp_acc.1/Atomicity					X			
fdp_rol.1/Atomicity					X			
fpt_fls.1/Operate		X						X
fru_flt.2								X
fpt_fls.1/SCP								X
fmt_lim.1								X
fmt_lim.2								X
fpt_php.3								X
fdp_itt.1								X
fpt_itt.1								X
fdp_ifc.1								X
fcs_rng.1				X				X
fpt_tst.1		X						X
fpt_emsec.1		X						X

Table 16 - Mapping of Security Functional Requirements (SFRs) on Security Functions (SFs)

5. QUALITY REQUIREMENTS

8.1 Revision History

Version	**Subject**
A	Initial Release (April 2015)

Table 17 - Revision History

6. ENVIRONMENTAL/ECOLOGICAL REQUIREMENTS

STMicroelectronics recommends viewing documents on the screen rather than printing to limit paper consumption.