Security Target Lite SmartCafe Expert V5.0

Version 1.0 / Status 27.10.2009

Author G&D/CSRD22
Status Open

Giesecke & Devrient GmbH
Prinzregentenstr. 159
Postfach 80 07 29
D-81607 München

© Copyright 2009 by
Giesecke & Devrient GmbH
Prinzregentenstr. 159
Postfach 80 07 29
D-81607 München

Contents

1 Introduction

1.1 ST Identification

Title: Security Target SmartCafe Expert V5.0
Reference: SmartCafe_Expert_ASE

Version Number: Version 1.0 / Status 27.10.2009

Origin: Giesecke & Devrient GmbH

Author: Dr. Ulrich Stutenbäumer

Compliant to: Java Card System Protection Profile, Version 1.0b, Standard 2.2 Configuration, August 2003: DCSSI-PP/0305 [JCSPP].

TOE: SmartCafe Expert V5.0

TOE documentation:

- Administrator Guidance Security Target SmartCafe Expert V5.0
- User Guidance Security Target SmartCafe Expert V5.0

HW-Part of TOE:

Configuration 1: NXP P5CD040V0B (BSI-DSZ-CC-0404-2007 *[NXP40]*)

Configuration 2: NXP P5CD080V0B (BSI-DSZ-CC-0410-2007 *[NXP80]*)

Configuration 3: NXP P5CD144V0B (BSI-DSZ-CC-0411-2007 *[NXP144]*)

1.2 ST Overview

The aim of this document is to describe the Security Target for **SmartCafe Expert V5.0**. In the following chapters **SmartCafe Expert V5.0** stands for the Target of Evaluation (TOE).

The related product is the **SmartCafe Expert V5.0 OS Java Card**.
In the following chapters SmartCafe Expert V5.0 OS Java Card stands for the product.

SmartCafe Expert V5.0 OS Java Card contains the TOE (see Figure 1 red dotted and straight red line) consisting of the:

- JCRE, JCVM, Java Card API´s, Remote Method Invocation (RMI), logical channels and applet and object deletion,
- the Card Manager,

- and the SCP (Smart Card Platform) consisting of IC (Integrated Circuit), OS (Chip Operating System) and DS (Chip Dedicated Software), *[NXP40]*, *[NXP80]*, *[NXP144]*

and depends on the secure IT environment (see Figure 1 blue dotted line) consisting of the off card Byte Code Verification.

Part of the SmartCafe Expert V5.0 OS Java Card is a fully interoperable VISA GlobalPlatform [VISA] compliant multi-application Java Card OS.

SmartCafe Expert V5.0 consists of the related software in combination with the underlying hardware ('Composite Evaluation').

The corresponding Security Target is based on the Java Card System – Standard 2.2 Configuration Protection Profile: DCSSI-PP/0305, Version 1.0b, August 2003.

This Security Target makes claims for formal conformance to this PP, as the ST fulfils all requirements of [JCSPP].

This ST even chooses a hierarchically higher augmentation of EAL4, in comparison to [JCSPP], by selecting ADV_IMP.2 and AVA_VLA.4.

Furthermore, parts of the IT security requirements were inspired from: Common Criteria Protection Profile — Machine Readable Travel Document with ICAO Application, Extended Access Control (PP-MRTD EAC), Version 1.1, 07.09.2006, BSI-PP-0026 [EAC] (see chapter 5.1.10).

This document describes:

- the Target of Evaluation (TOE): TOE Description
- the security environment of the TOE: TOE security environment
- the security objectives of the TOE and its environment: Security objectives
- the TOE security functional and assurance requirements: TOE security functional requirements, TOE security assurance requirements

The assurance level for the TOE is CC **EAL4 augmented**.

The minimum strength level for the TOE security functions is **high (SOF high)**.

1.2.1 CC Conformance

This TOE is compliant to Common Criteria V2.3 (ISO 15408) as follows:

- Part 2 conformant extended by FCS_RND.1, FMT_LIM.1, FMT_LIM.2 and FPT_EMSEC.1.

- Part 3 conformant,
- Package conformant to EAL4 augmented with:
    - **AVA_VLA.4** Vulnerability Assessment - Vulnerability Analysis - Highly resistant, and
    - **ADV_IMP.2** Development – Implementation Representation – Implementation of the TSF.

The minimum strength level for the TOE security functions is **SOF high**.

1.2.2 Sections Overview

Section 1 provides the introductory material for the Security Target.

Section 2 provides general purpose and TOE description.

Section 3 provides a discussion of the expected environment for the TOE. This section also defines the set of threats that are to be addressed by either the technical countermeasures implemented in the TOE hardware, the TOE software, or through the environmental controls.

Section 4 defines the security objectives for both the TOE and the TOE environment.

Section 5 contains the functional requirements and assurance requirements derived from the Common Criteria (CC), Part 2 [CC2] and Part 3 [CC3], which must be satisfied.

Section 6 contains the TOE Summary Specification.

Section 7 provides that there is a compliance claim to a PP.

Section 8 provides a rationale to explicitly demonstrate that the information technology security objectives satisfy the policies and threats. Arguments are provided for the coverage of each policy and threat. The section then explains how the set of requirements are complete relative to the objectives, and that each security objective is addressed by one or more component requirements. Arguments are provided for the coverage of each objective. Next section 8 provides a set of arguments that address dependency analysis, strength of function issues, and the internal consistency and mutual supportiveness of the protection profile requirements

Section 9 provides information on used references and of frequently used acronyms.

Section 10 provides a glossary.

1.3 Typographic Conventions

- *This typeface* is used to highlight those words that appear in the Appendix: Glossary. Example: *applet*.
- **<u>This typeface</u>** is used to highlight assignments and selections for SFRs completed by the ST author.

- **This typeface** or ***this typeface*** is used to highlight assignments and selections for SFRs defined in the PP.

1.4 Change History

Version	Date	Changes	Responsible
1.0	27.10.09	Final Version	stut

1.5 Figures

1.6 Tables

1.7 Application Notes of the PP

When applicable the application notes of the PP are discussed in notes.

2 TOE Description

2.1 TOE abstract

The TOE under evaluation is **SmartCafe Expert V5.0,** containing the smart card platform[1] (SCP in Figure 1).

Parts of the TOE are the Java Card Runtime Environment (JCRE), the Java Card Virtual Machine (JCVM) and the Java Card API (see dotted red line in Figure 1). Additional to the Standard 2.2 configuration of the Java Card System Protection Profile the Card Manager and the Smart Card Platform was also chosen as part of the TOE for this security target.

The TOE depends on its IT environment that is the bytecode verifier (see blue line in Figure 1).

The final product **SmartCafe Expert V5.0 OS Java Card** contains Java and no native applications. However, there are vendor-specific libraries present on the card that are available to applets. These libraries contain native code and are not part of the TOE.

The SmartCafe Expert V5.0 OS Java Card contains **Smart Card Embedded Software** that is hosted on a certified Smart Card Integrated Circuit (IC) with comparable level to the current TOE evaluation and provides a platform for financial applications e.g.

The product (**SmartCafe Expert V5.0 OS Java Card**) containing the TOE is based on and designed to be compliant to the following specifications:

- The Java Card specification (see: [JCAPI22], [JCRE22], [JCVM22])[2];
- Visa GlobalPlatform 2.1.1 Card Implementation Requirements [VISA].

These de facto standards are aimed at defining a framework with which Applications can be developed, managed and used on a Java Card Platform Embedded Software.

SmartCafe Expert V5.0 applies:

- The chip's security requirements for the OS (Chip Operating System) and DS (Chip Dedicated Software) of **NXP P5CD040/080/144**
- Java Card Development kit 2.2.2

[1] The smart card platform consists of the NXP IC, the OS (partial native, partially written in Java code) and native DS.

[2] The TOE is compliant to both JCS specifications: 2.1 and 2.2.2.

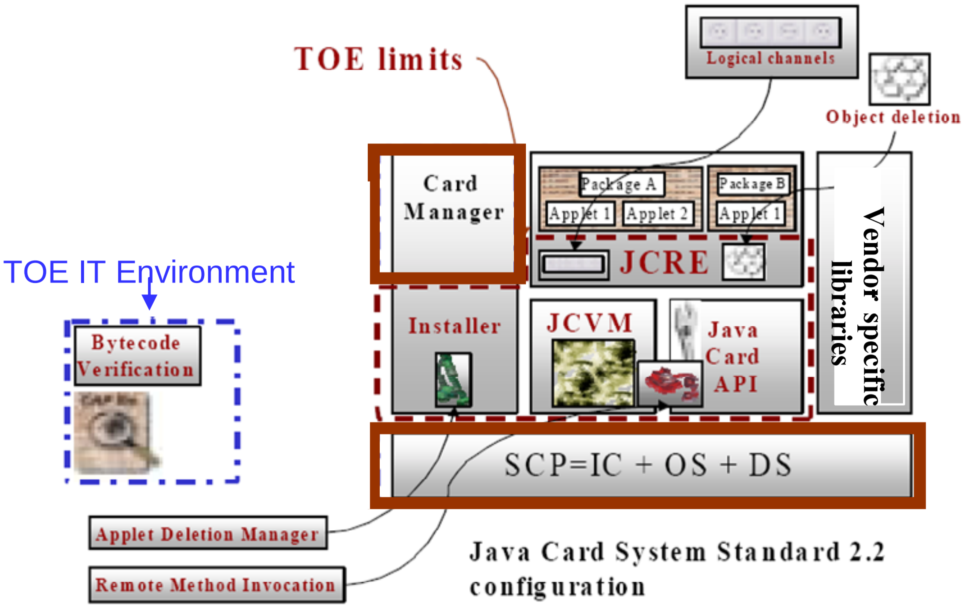


Figure 1 TOE Limits (dotted and straight red line)

SCP (Smart Card Platform), IC (Integrated Circuit), OS (Chip Operating System), DS (Chip Dedicated Software). Contrary to the Java Card System Standard 2.2 configuration native Applications are not part of the product except vendor-specific libraries.

Each TOE module under evaluation (inside dotted redline in figure 1) and the OS of the SCP is developed by Giesecke & Devrient and based on the previous specified specifications.

The TOE includes the Smart Card Platform that is the micro-controller (as for a composite evaluation the other TOE components comply with the certified chip’s security requirements), the firmware (dedicated software (DS)), the Card Manager and the operating system (OS). The Native Applications, the Bytecode Verification and the Application layer are not part of the TOE.

Note: Native Applications are also not part of the product SmartCafe Expert V5.0 OS Java Card except vendor-specific libraries. The loading of native applications on the SmartCafe Expert V5.0 OS Java Card is not permitted.

Note: The TOE uses information provided by the micro-controller to detect attacks.

2.2 Product Type

Part of the SmartCafe Expert V5.0 OS **Java Card** is a fully interoperable VISA GlobalPlatform compliant multi-application Java Card OS.

SmartCafe Expert V5.0 OS Java Card offers Java Card technology [JCAPI22], [JCRE22], [JCVM22] and VISA GlobalPlatform 2.1.1 services [VISA] to applets on the chip such as financial applets.

2.2.1 Features of SmartCafe Expert V5.0 OS Java Card

Features of SmartCafe Expert V5.0 OS Java Card include:

- Compliant with Java Card™ release 2.2.2
- Convenient object-oriented programming in high-level language Java™
- Hardware random number generator and ANSI X9.31 software random number generator
- DES/3DES [DES], RSA [RSA] and AES [AES] cryptography
- Security-relevant methods are DPA/SPA secured and DFA resistant
- Compliant to ISO 7816 Parts 1-9 [ISO]
- ISO 14443 Type A

2.2.2 Physical scope of TOE

The TOE consists of the following parts:

- NXP P5CD040/080/144 and Dedicated Software
- Java Card Runtime Environment (JCRE)
- Java Card Virtual Machine (JCVM)
- Java Card API
- On-card Installer
- Applet Deletion Manager
- Remote Method Invocation (RMI)
- Card Manager
- Smart Card OS

2.2.3 Logical scope of TOE

The TOE provides the following services:

- Logical Channels
- Object Deletion
- atomistic rollback and optimistic backup

- firewall access control
- 3-DES (2-key/3-key) signature by MAC computation
- RSA and AES cipher/decipher
- integrity check of check summed data
- secure state of information
- non-observability of operations on sensitive information
- unavailability of previous information content
- secure installation of post-issuance applications on to the card
- secure post-issuance deletion of previously installed applets

2.2.4 Delivery scope of TOE

The TOE ROM code on the IC and additional mask keys are delivered to the initialisation site. The initialisation file is then securely loaded on the Smart Card at the initialisation site.

The initialisation and the personalisation process are out of scope of this evaluation.

The TOE's delivery scope beside the initialisation file and the mask keys consists out of the following documentation for the Smart Card issuer and applet developer:

- Administrator Guidance Security Target SmartCafe Expert V5.0
- User Guidance Security Target SmartCafe Expert V5.0

2.2.5 Non TOE Features of SmartCafe Expert V5.0 OS Java Card

The following features of SmartCafe Expert V5.0 OS Java Card are not part of the TOE:

- Biometric API with ISO 19794 fingerprint on-card matching
- Implementation according to GlobalPlatform Card Specification 2.1.1 [VISA]
- DSA and elipitic curve cryptography
- MIFARE
- SHA256

2.3 TOE life cycle

The main part of the following description (chapter 2.3 to 2.6) is taken from the corresponding Protection Profile [JCSPP], chapter 2.4.2. Changes were made when necessary to reflect the present TOE that is augmented compared to the TOE of the Standard 2.2 Configuration of the PP [JCSPP]. The TOE is extended by the Card Manager and the Smart Card Platform.

2.3.1 The TOE in the Life Cycle of the Smart Card

Following the CC, we separate the TOE environment into two parts: the IT environment and the non-IT environment. As seen in the preceding sections, the TOE is intended to be part of an IT product embedded in a smart card; due to specific development and installation processes of the smart card industry, these (the TOE's development and installation) are not separable from that of the other IT components of the smart card. This development phase constitutes the main part of the non-IT environment of the TOE.

The rest of this section is inspired by [PP0010], as we assume that JCRE is part of the embedded software (ES), so the same development rules shall apply.

The life cycle of the TOE, which is only a part of the smart card life cycle, can be reduced to the **three stages** pictured in Figure 2, called **Development**, **Production & Personalization**, and **Usage**.

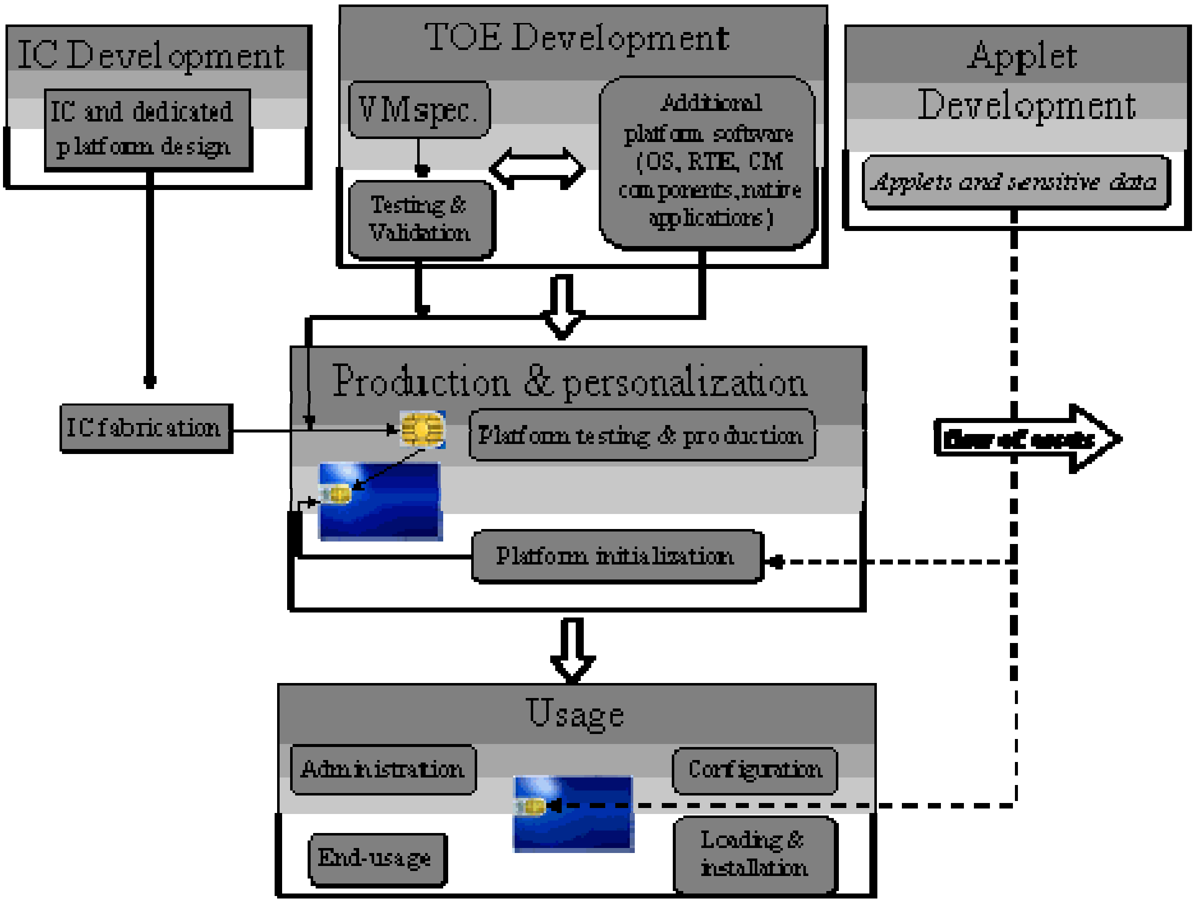

IC Development
IC and dedicated platform design
TOE Development
VM spec.
Testing & Validation
Additional platform software (OS, RTE, CM components, native applications)
Applet Development
Applets and sensitive data
IC fabrication
Production & personalization
Platform testing & production
Platform initialization
flow of assets
Usage
Administration
Configuration
End-usage
Loading & installation

Figure 2 Smart Card Product Life Cycle

2.3.1.1 TOE Development & Production Environments

The development and production of the TOE is carried out during the first and second stages. To ensure security, the environment in which the development takes place must be made secure with controllable accesses and traceability. Furthermore, it is important that every authorized personnel involved fully understands the importance and the rigid implementation of defined security procedures.

The development begins with the TOE specification. All parties in contact with sensitive information are required to abide by Non-Disclosure Agreements.

Development of the TOE then follows. The engineers use a secure computer system (preventing unauthorized access) to make their specifications, design, development and generation of the product. Storage of sensitive documents, databases on tapes, diskettes are in appropriately locked cupboards/safe. The disposal of unwanted data (complete electronic erasures) and documents (like shredding) is also of great importance. Testing, integration and validation of TOE components then take place. This phase consists in the collection of all software modules and the execution/test of this software on an emulator or on a simulator of the (DS & IC) layer.

When these are done offsite, they must be transported and worked out in a secure environment with accountability and traceability of all components. During the electronic transfer of sensitive data, procedures must be established to ensure that the data and programs reach the expected destination and are not accessible at intermediate stages (stored on a buffer server where system administrators make backup copies). Should the integration tests be successful, the ROM code is delivered to the IC manufacturer.

During **the production** stage the TOE is used in the IC Packaging, smart card Finishing process and the test environments. Everyone involved in such operations shall fully understand the importance of security procedures. Moreover, the environment in which these operations take place must be secured. Sensitive information (on tapes, disks or diskettes) is stored in an appropriately locked cupboard/safe. Also of paramount importance is the disposal of unwanted data (like complete electronic erasures) and documents (for instance, shredding). Personalization then occurs that is, the embedder introduces data for configuration and initialization of software components, namely the OS, the *Java Card System*, the *SCP*, and applications. At the end of the second stage, the TOE is fully functional.

Adequate control procedures are necessary to account for all products at all stages. These must be transported and manipulated in a secure environment with accountability and traceability of all (good and bad) products.

2.3.1.2 TOE Final Environment

The **third stage** is the end usage time of the TOE.

Once the previous stage is over, the loading and installation of applications, and configuration (initialization) of user data (like user PIN) is done. The card is finally issued to the end user (card holder).

The main users of the TOE at this time are the applications, either pre-installed or loaded. The end user environment thus covers a wide spectrum of very different functions.

However, we can define the IT environment during this phase: the bytecode verifier.

The TOE communicates with the CAD through the card manager.

During normal usage, the card is inserted in a CAD[3], starting up the card manager and *JCRE*. The session is an exchange of APDU commands between the CAD and the card manager, the card manager and the *JCRE* and, ultimately, the *JCRE* and some *applet*.

Finally, that loading issue leads us to another entity, which appears in Figure 3, the CAP file verifier (also known as “bytecode verifier”, or, shortly, the *BCV*). The verifier is located off–card.

During the **usage phase** the administrator can delete applets by the applet deletion manager in a secure way.

2.4 TOE intended usage

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

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

Applications installed on a Java Card platform can be selected for execution when the card is inserted into a card reader. In some configurations of the TOE, the card reader may also be used to enlarge or restrict the set of applications that can be executed on the Java Card platform according to a well-defined card management policy.

Notice that these applications may contain other confidentiality (or integrity) sensitive data than usual cryptographic keys and PINs; for instance, passwords or pass-phrases are as confidential as the PIN, and the balance of an electronic purse is highly sensitive with regard to arbitrary modification (because it represents real money).

So far, the most important applications are:

– Financial applications, like Credit/Debit ones, stored value purse, or electronic commerce, among others.

[3] In this ST the card is able to communicate in contactless or contact mode.

- Transport and ticketing, granting pre-paid access to a transport system like the metro and bus lines of a city.
- Personal identification, for granting access to secured sites or providing identification credentials to participants of an event.
- 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.

The version 2.2 of the Java Card platform (“Java Card System 2.2”) introduces several novelties that extend the domain of applications of the Java Card platform and ensures its compatibility with the industrial state-of-art standards. One of those features is the possibility of having more than one *applet* selected for execution at a time, which is intensively used in identity modules of mobile phone applications. A Java Card platform implementing this feature is said to support “logical channels”.

Java Card System 2.2 also provides *applet* deletion, which enables the fine tuning of open card management. This typically impacts the loyalty applications, which are obvious candidates for post-issuance downloading and removal of applications.

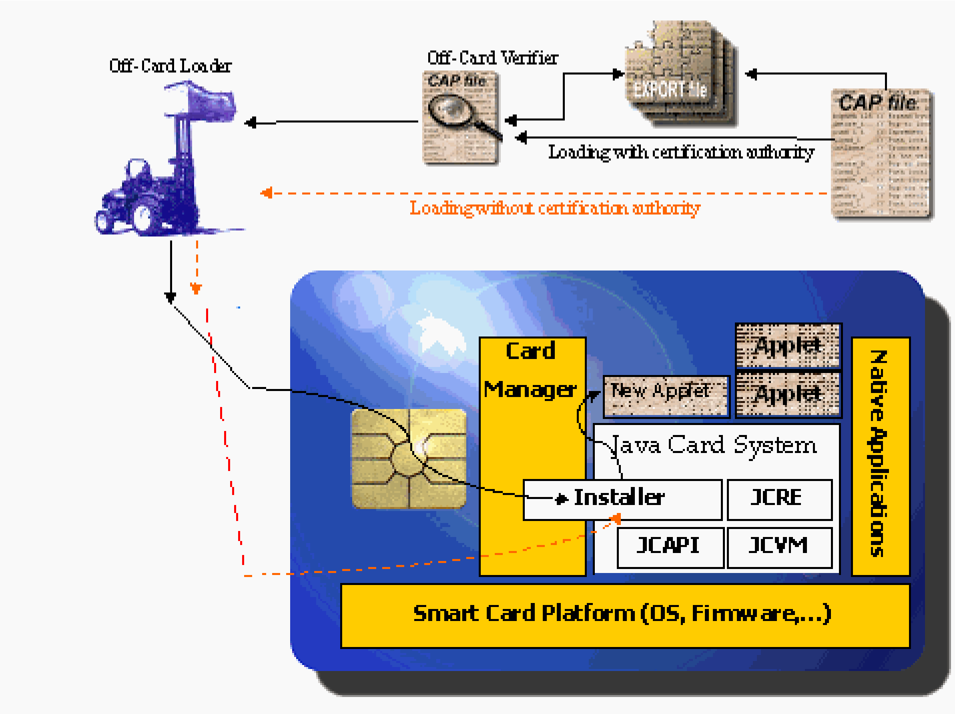


Figure 3 Usage environment of the TOE

2.5 Scope of Evaluation

Part of the scope of the TOE is the *Java Card System*. The integrated circuit, the operating system and the dedicated software of the smart card (SCP) are also part of the TOE. But any piece of native code that does not contribute to its implementation is not part of the TOE, like a native application embedded together with Java Card applications. Regarding the components of the TOE, one may distinguish the software part, consisting of the Java Card System and the OS, from the Smart Card as a hardware component. Both parts are included in the evaluation, by forming a composite TOE. The good working order of this composite TOE much depends on the way the software parts are handled during the manufacturing process of the card (for instance, how they are embedded into the card).

Thus the scope of evaluation actually includes more than the TOE itself. The Common Criteria acknowledges this situation, allowing security requirements applying to the development and construction of the TOE, stated in several SARs (security assurance requirements), particularly those from the ADO (delivery) and ACM (configuration) classes [CC3].

Let us also remark that the code of the applets is not part of the code of the TOE, but just data managed by the TOE. Applets are only considered in their CAP format, and the process of compiling the source code of an application and converting it into the CAP format does not regard the TOE or its environment. On the contrary The process of verifying applications in its CAP format and loading it on the card is a crucial part of the TOE environment and plays an important role as a complement of the TSFs included in the configuration. The PPs assume that the loading of applications pre-issuance is made in a secure environment. For post-issuance phases, the card will need to protect itself[4].

Native applications[5] may be placed into the card not through the *installer* component of the *Java Card System*, but by directly embedding them into the IC during the fabrication of the smart card, along with that of the *Java Card System*. This is the usual way to have native methods installed, but the process is not limited to them, and *applet*s and API *package*s may also be installed at a time where the TOE is not yet operational. This also advocates for including several security assurance requirements on the life cycle of the smart card, since native applications are not under the control of the *Java Card System.*

The TOE is an appropriate Embedded Software to implement the *Card issuer*'s policy in order to provide a JCP which can load and install Java Card applications during

[4] This protection is to be on the behalf of the card manager (protection by applying the card manager keys).

[5] Native Applications are also not part of the SmartCafe Expert V5.0 OS Java Card. The loading of native applications on the SmartCafe Expert V5.0 OS Java Card is not permitted.

installation and usage phases and run the installed applets in the usage phase (see Figure 3).

2.6 Product Rationale

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

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

The *applet firewall* is considered as the most important security feature. It enables complete isolation between *applet*s or controlled communication through additional mechanisms that allow them to share objects when needed. The *JCRE* allows such sharing using the concept of “shareable interface objects” (*SIO*) and **`static public`** variables. The *JCVM* should ensure that the only way for *applet*s to access any resources are either through the *JCRE* or through the Java Card API (or other vendor-specific APIs). This objective can only be guaranteed if *applet*s are correctly typed (all the “must clauses” imposed in chapter 7 of [JCVM22] on the bytecodes and the correctness of the *CAP* file format are satisfied).

Secure loading of Java Card packages can be achieved by off card byte code verification of the packages prior to the loading on to the card with Sun certified bytecode verification tools. Secure installation and deletion of applets are protected by the card manager that can prevent any unauthorised loading and deletion of applications on the card.

3 TOE security environment

3.1 Security Aspects

Security aspects are intended to define the main security issues that are to be addressed in the PP, 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. We will define hereafter the following aspect:

3.1.1 Confidentiality

3.1.1.1 #.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.

3.1.1.2 #.CONFID-JCS-CODE

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

3.1.1.3 #.CONFID-JCS-DATA

Java Card System data must be protected against unauthorized disclosure. This concerns logical attacks at runtime in order to gain a read access to Java Card System data. Java Card System data includes the data managed by the Java Card runtime environment, the virtual machine and the internal data of Java Card API classes as well.

3.1.2 Integrity

3.1.2.1 #.INTEG-APPLI-CODE

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

3.1.2.2 #.INTEG-APPLI-DATA

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

3.1.2.3 *#.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.

3.1.2.4 *#.INTEG-JCS-DATA*

Java Card System data must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to Java Card System data. Java Card System data includes the data managed by the Java Card runtime environment, the virtual machine and the internal data of Java Card API classes as well.

3.1.3 Unauthorized Executions

3.1.3.1 *#.EXE-APPLI-CODE*

Application (byte)code must be protected against unauthorized execution. This concerns

(1) invoking a method outside the scope of the visibility rules provided by the public/private access modifiers of the Java programming language ([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*.

3.1.3.2 *#.EXE-JCS-CODE*

Java Card System (byte)code must be protected against unauthorized execution. *Java Card System* (byte)code includes any code of the *JCRE* or *API*. This concerns

(1) invoking a method outside the scope of the visibility rules provided by the public/private access modifiers of the Java programming language ([JAVASPEC];

(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*.

3.1.3.3 *#.FIREWALL*

The *Java Card System* shall ensure controlled sharing of class instances[6], and isolation of their data and code between *package*s (that is, controlled execution contexts). **(1)** An *applet* shall neither read, write nor compare a piece of data belonging to an *applet* that is not in the same context, nor execute one of the methods of an applet in another context without its authorization.

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

3.1.3.4 #.NATIVE

Because the execution of native code is outside of the TOE Scope Control (TSC), it must be secured so as to not provide ways to bypass the TSFs. No untrusted native code may reside on the card. Loading of native code, which is as well outside the TSC, is submitted to the same requirements. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

3.1.4 Bytecode Verification

3.1.4.1 #.VERIFICATION

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

3.1.4.1.1 CAP File Verification

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

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

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

3.1.4.1.2 Integrity and Authentication

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

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.

3.1.4.1.3 Linking and Verification

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

3.1.5 Card Management

3.1.5.1 #.CARD-MANAGEMENT

(1) The card manager (CM) shall control the access to card management functions such as the installation, update or deletion of *applet*s. **(2)** The card manager shall implement the card issuer’s policy on the card.

3.1.5.2 #.INSTALL

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

3.1.5.3 #.SID

(1) Users and subjects of the TOE must be identified.

(2) The identity of sensitive users and subjects associated with administrative and privileged roles must be particularly protected; this concerns the *JCRE*, the applets registered on the card, and especially the *default applet* and the *currently selected applet* (and all other active applets in Java Card System 2.2). A change of identity, especially standing for an administrative role (like an *applet* impersonating the *JCRE*), is a severe violation of the TOE Security Policy (TSP). Selection controls the access to any data exchange between the TOE and the *CAD* and therefore, must be protected as well. The loading of a *package* or any exchange of data through the *APDU buffer* (which can be accessed by any applet) can lead to disclosure of keys, application code or data, and so on.

3.1.5.4 #OBJ-DELETION

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

3.1.5.5 #DELETION

Deletion of applets must be secure. **(1)** Deletion of installed *applet*s (or *package*s) should not introduce security holes in the form of broken references to garbage collected code or data, nor should they alter integrity or confidentiality of remaining *applet*s. The deletion procedure should not be maliciously used to bypass the TSFs. **(2)** Erasure, if deemed successful, shall ensure that any data owned by the deleted *applet* is no longer accessible (shared objects shall either prevent deletion or be made inaccessible). A deleted *applet* cannot be selected or receive APDU commands. Package deletion shall make the code of the package no longer available for execution.**(3)** Power failure or other failures during the process shall be taken into account in the implementation so as to preserve the TSPs. This does not mandate, however, the process to be atomic. For instance, an interrupted deletion may result in the loss of user data, as long as it does not violate the TSPs.

The deletion procedure and its characteristics (whether deletion is either physical or logical, what happens if the deleted application was the *default applet*, the order to be observed on the deletion steps) are implementation-dependent. The only commitment is that deletion shall not jeopardize the TOE (or its assets) in case of failure (such as power shortage).

Deletion of a single *applet* instance and deletion of a whole *package* are functionally different operations and may obey different security rules. For instance, specific *package*s can be declared to be undeletable (for instance, the Java Card API *package*s), or the dependency between installed *package*s may forbid the deletion (like a *package* using super classes or super interfaces declared in another package).

3.1.6 Services

3.1.6.1 #.ALARM

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

3.1.6.2 #.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.

3.1.6.3 #.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*.

3.1.6.4 #.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.

3.1.6.5 #.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.

3.1.6.6 #.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

3.1.6.7 #.SCP

The smart card platform must be secure with respect to the TSP. Then:

(1) After a power loss or sudden card removal prior to completion of some communication protocol, the *SCP* will allow the Java Card System on the next power up to either complete the interrupted operation or revert to a secure state.

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

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

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

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

(6) It safely transmits low–level exceptions to the Java Card System (arithmetic exceptions, checksum errors), when applicable.
We finally require that

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

3.1.6.8 #.TRANSACTION

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

3.2 Assets

Assets are security–relevant elements to be directly protected by the TOE. Confidentiality of assets is always intended with respect to un-trusted people or software, as various parties are involved during the first stages; 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 source code (one asset) or compiled code (another asset), and may exist in various formats (digital supports, printed paper) at different stages of its development. 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 weighs on it.

3.2.1 User data

3.2.1.1 D.APP_CODE

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

3.2.1.2 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.

3.2.1.3 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.

3.2.1.4 D.PIN

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

3.2.1.5 D.APP_ KEYs

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

3.2.2 TSF data

3.2.2.1 D.JCS_CODE

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

3.2.2.2 D.JCS_ DATA

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

3.2.2.3 D.SEC_DATA

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

3.2.2.4 D.API_DATA

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

3.2.2.5 D.JCS_KEYs

Cryptographic keys used when loading a file into the card. To be protected from unauthorized disclosure and modification.

3.2.2.6 D.CRYPTO

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

3.3 User & Subjects

The users of the TOE include people (like the end-user, applet developer) and hardware (like the *CAD* where the card is inserted). Subjects are active components of the TOE that (essentially) act on the behalf of *users*.

3.3.1 S.PACKAGE

*Package*s used on the Java Card platform that act on behalf of the applet developer. These subjects are involved in the ***FIREWALL security policy*** defined in § 5.1.1.1 of [JCSPP] and they should be understood as instances of the subject *S.PACKAGE*.

3.3.2 S.JCRE

The *JCRE*, which acts on behalf of the card issuer. This subject is involved in several of the security policies defined in this document and is always represented by the subject *S.JCRE* (defined in 5.1.4.1).

3.3.3 S.CRD

The *installer*, which acts on behalf of the card issuer. This subject is involved in the loading of *package*s and installation of *applet*s. It could play the role of the on-card entity in charge of package loading, which is involved in the ***PACKAGE LOADING security policy*** defined in § 5.1.8 of [JCSPP] and is represented by the subject *S.CRD*.

3.3.4 S.ADEL

The *applet deletion manager*, if the configuration contains such components, which also acts on behalf of the card issuer. This subject is involved in the ***ADEL security policy*** defined in § 5.1.4.1 of [JCSPP] and is represented by the subject *S.ADEL*.

3.3.5 S.BCV

The bytecode verifier (BCV), which acts on behalf of the verification authority. This subject is involved in the PACKAGE ***LOADING security policy*** defined in § 5.1.8 [[JCSPP]] and is represented by the subject *S.BCV*.

3.3.6 S.CAD

The *CAD* is involved in the **JCRMI policy** defined in § 5.1.5.1 [[JCSPP]] and is represented by the subject *S.CAD*.

3.4 Assumptions

This section introduces the assumptions made on the environment of the TOE for each of the configurations considered in this document.

3.4.1 A.NATIVE

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

3.4.2 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.

3.4.3 A.APPLET

Applets loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM22], §3.3) outside the API.

3.5 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.

3.5.1 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 DP analysis. That also includes the modification of the runtime execution of Java Card System or SCP software through alteration of the intended execution order of (set of) instructions through physical tampering techniques. This threatens all the identified assets. This threat refers to #.SCP.7, and all aspects related to confidentiality and integrity of code and data.

3.5.2 T.CONFID-JCS-CODE

The attacker executes an application without authorization to disclose the Java Card System code. See #.CONFID-JCS-CODE for details. Directly threatened asset(s): D.JCS_CODE.

3.5.3 T.CONFID-APPLI-DATA

The attacker executes an application without authorization 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.

3.5.4 T.CONFID-JCS-DATA

The attacker executes an application without authorization to disclose data belonging to the Java Card System. See #.CONFID-JCS-DATA for details. Directly threatened asset(s): D.API_DATA, D.SEC_DATA, D.JCS_DATA D.JCS_KEYs and D.CRYPTO.

3.5.5 T.INTEG-APPLI-CODE

The attacker executes an application to alter (part of) its own or another application’s code. See #.INTEG-APPLI-CODE for details. Directly threatened asset(s): D.APP_CODE

3.5.6 T.INTEG-JCS-CODE

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

3.5.7 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.

3.5.8 T.INTEG-JCS-DATA

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

3.5.9 T.SID.1

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

3.5.10 T.SID.2

The attacker modifies the identity of the privileged roles. 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).

3.5.11 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.

3.5.12 T.EXE-CODE.2

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

3.5.13 T.NATIVE

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

3.5.14 T.RESOURCES

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

3.5.15 T.INTEG-APPLI-CODE.2

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**.

3.5.16 T.INTEG-APPLI-DATA.2

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_APP_KEYs** and **D.JCS_KEYs.**

3.5.17 T.INSTALL

The attacker fraudulently **installs** post-issuance of an *applet* on the card. This concerns either the installation of an unverified *applet* or an attempt to induce a malfunction in the TOE through the installation process. See #.*INSTALL* for details.

Directly threatened asset(s): **D.SEC_DATA** (any other asset may be jeopardized should this attack succeed, depending on the virulence of the installed application).

3.5.18 T.EXE-CODE-REMOTE

The attacker performs an unauthorized **remote execution** of a **method** from the *CAD*. See #.*EXE-JCS-CODE* and #.*EXE-APPLI-CODE* for details.

Directly threatened asset(s): **D.APP_CODE**.

This threat concerns version 2.2 of the Java Card System remote method invocation features, which allow external users (that is, other than on-card applets) to trigger the execution of code belonging to an on-card applet. On the contrary, T.EXE-CODE.1 is restricted to the applets under the TSC.

3.5.19 T.DELETION

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

3.5.20 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** & **D.APP_KEYs** .

3.5.21 T.RND

The following threat was taken over from [SCPP].

Deficiency of Random Numbers

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

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

Here the attacker is expected to take advantage of statistical properties of the random numbers generated by the TOE without specific knowledge about the TOE's generator. Malfunctions or premature ageing are also considered which may assist in getting information about random numbers.

3.5.22 T.LEAKAGE

The following threat was not taken from a Protection Profile.

The attacker exploits information which is leaked from the TOE to disclose the confidential primary assets.

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

3.5.23 T.CHIP

The following threat was not taken from a Protection Profile.

The attacker delivers Smartcard ICs to the TOE Manufacturer that are not correctly tested and pre-personalised by the Chip Manufacturer.

3.6 Organisational security policies

This configuration has only one organizational security policy:

3.6.1 OSP.VERIFICATION

This policy shall ensure the adequacy 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.

4 Security objectives

4.1 Security objectives for the TOE

4.1.1 O.SID

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

4.1.2 O.OPERATE

The TOE must ensure continued correct operation of its security functions. See #.OPERATE for details.

4.1.3 O.RESOURCES

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

4.1.4 O.FIREWALL

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

4.1.5 O.NATIVE

The only means that the JCVM 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.

4.1.6 O.REALLOCATION

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

Note: To be made unavailable means to be physically erased with a default value. Except for local variables that do not correspond to method parameters, the default values to be used are specified in [JCVM22].

4.1.7 O.SHRD_VAR_CONFID

The TOE shall ensure that any data container that is shared by all applications is always cleaned after the execution of an application. Examples of such shared containers are the APDU buffer, the byte array used for the invocation of the process method of the selected applet, or any public global variable exported by the API.

4.1.8 O.SHRD_VAR_INTEG

The TOE shall ensure that only the currently selected application may grant write access to a data memory area that is shared by all applications, like the APDU buffer, the byte array used for the invocation of the process method of the selected applet, or any public global variable exported by the API. Even though the memory area is shared by all applications, the TOE shall restrict the possibility of getting a reference to such memory area to the application that has been selected for execution. The selected application may decide to temporarily hand over the reference to other applications at its own risk, but the TOE shall prevent those applications from storing the reference as part of their persistent states.

4.1.9 O.ALARM

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

4.1.10 O.TRANSACTION

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

4.1.11 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.

4.1.12 O.PIN-MNGT

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

Note: The ways PIN objects are stored and managed in the memory of the smart card are carefully considered, and this applies to the whole object rather than the sole value of the PIN.

4.1.13 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.

Note: O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION and O.CIPHER are actually provided to applets in the form of Java Card APIs. Vendor-specific libraries are present on the card and are available to applets; those are built independently of the Java Card API. Those libraries contain native code and are not part of the TOE.

4.1.14 O.INSTALL

The TOE shall ensure that the installation of an *applet* is safe. See #*.INSTALL*.

4.1.15 O.LOAD

The TOE shall ensure that the loading of a *package* into the card is safe.

Note: Usurpation of identity resulting from a malicious installation of an applet on this card may also be the result of perturbing the communication channel linking the CAD and the card. Even if the CAD is placed in a secure environment, the attacker may try to capture, duplicate, permute or modify the packages sent to the card. He may also try to send one of its own applications as if it came from the card issuer. Thus, this objective is intended to ensure the integrity and authenticity of loaded CAP files.

4.1.16 O.DELETION

The TOE shall ensure that both *applet* and *package* deletion are safe. See #*.DELETION* for details.

4.1.17 O.OBJ-DELETION

The TOE shall ensure the object deletion shall not break references to objects. See #*.OBJ-DELETION* (p. 26) for further details.

4.1.18 O.REMOTE

The TOE shall provide a means to restrict remote access from the *CAD* to the services implemented by the applets on the card. This particularly concerns the *RMI* services introduced in version 2.2 of the Java Card platform.

4.1.19 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. See #.SCP.1.

4.1.20 O.SCP.SUPPORT

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

4.1.21 O.SCP.ICThe SCP shall possess IC security features. See #.SCP.7.

4.1.22 O.CARD-MANAGEMENT

The card manager shall control the access to card management functions such as the installation, update or deletion of applets. It shall also implement the card issuer’s policy on the card.

The card manager is an application with specific rights, which is responsible for the administration of the smart card. This component will in practice be tightly connected with the TOE, which in turn shall very likely rely on the card manager for the effective enforcing of some of its security functions. Typically the card manager shall be in charge of the life cycle of the whole card, as well as that of the installed applications (*applet*s). The card manager should prevent that card content management (loading, installation, deletion) is carried out, for instance, at invalid states of the card or by non-authorized actors. It shall also enforce security policies established by the card issuer.

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

4.1.23 O.RND

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

Random Numbers

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

The TOE will ensure that no information about the produced random numbers is available to an attacker since they might be used for instance to generate cryptographic keys.

4.1.24 O.SIDE_CHANNEL

The following security objective is not taken from a Protection Profile.

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 to avoid interpretations of signals extracted from the hardware part of the TOE (Power Supply, Electro Magnetic emissions, e.g.).

4.1.25 O.CHECK_INIT

The following security objective is not taken from a Protection Profile.
To ensure the receipt of the correct TOE from the IC Manufacturer by the TOE Manufacturer, the SCP shall check a sufficient part of the pre-personalisation data. This shall include at least the FabKey Data that is agreed between the TOE Manufacturer and the Chip Manufacturer.

4.2 Security objectives for the environment

These environmental objectives shall be met by IT security requirements.

4.2.1 OE.NATIVE

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

4.2.2 OE.APPLET

No *applet* loaded post-issuance shall contain native methods.

4.2.3 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.

5 IT security requirements

5.1 TOE security functional requirements

The following table displays the relationship between the chosen configuration of the PP [JCSPP] and the actual ST and the groups that are defined in the PP.

PP Group Name	Java Card System Standard 2.2 Configuration	This ST
Core (*CoreG*)	TOE	TOE
Smart card platform (*SCPG*)	IT	TOE
Installer (*InstG*)	TOE	TOE
RMI (*RMIG*)	TOE	TOE
Logical channels (*LCG*)	TOE	TOE
Object deletion (*ODELG*)	TOE	TOE
Bytecode verification (*BCVG*)	IT	IT
Applet deletion (*ADELG*)	TOE	TOE
Secure carrier (*CarG*)	TOE	TOE
Card manager (*CMGRG)*	IT	TOE

Table 1 Relationship between Groups and PP Configuration and the ST

This section defines the functional requirements for the TOE using only functional requirements components drawn from the CC part 2 except four additional SFR (FCS_RND.1, FMT_LIM.1, FMT_LIM.2, FPT_EMSEC.1) not contained in CC part 2. The functional requirements were taken from [JCSPP] (sections 5.1.1- 5.1.9) and newly defined (section 5.1.10) were taken from [EAC].

5.1.1 CoreG Security Functional Requirements

5.1.1.1 Information Flow Control Policy

FDP_IFC.1/JCVM Subset information flow control

5.1.1.1.1 FDP_IFC.1.1/JCVM

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

Subjects[7] (prefixed with an "*S*") and information (prefixed with an "*I*") covered by this policy are:

Subject/Information	***Description***
S.LOCAL	Operand stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.
S.MEMBER	Any object's field, **static** field or array position.
I.DATA	JCVM Reference Data: *objectref addresses of temporary JCRE Entry Point objects and global arrays*.

Table 2 Subjects and information for JCVM information flow control SFP

There is a unique operation in this policy:

Operation	Description
OP.PUT(S_1, S_2, I)	Transfer a piece of information *I* from S_1 to S_2.

Table 3 Operation for JCVM information flow control SFP

Note: References of temporary *JCRE entry points*, which cannot be stored in *class* variables, instance variables or array components, are transferred from the internal memory of the *JCRE* (TSF data) to some stack through specific APIs (*JCRE* owned exceptions) or *JCRE* invoked methods (such as the **process(APDU apdu)**); these are causes of *OP.PUT(S_1,S_2,I)* operations as well.

FDP_IFF.1/JCVM Simple security attributes

5.1.1.1.2 FDP_IFF.1.1/JCVM

Changed by [CCFI_104] to:

The TSF shall enforce the ***JCVM information flow control SFP*** based on the following types of subject and information security attributes: ***S.LOCAL, S.MEMBER and I.DATA*** and the ***currently active context.***

5.1.1.1.3 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) is allowed if and only if the active context is "JCRE"; other OP.PUT operations are allowed regardless of the active context's value.

5.1.1.1.4 FDP_IFF.1.3/JCVM

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

The TSF shall enforce **the additional information flow control SFP rules: none**

5.1.1.1.5 FDP_IFF.1.4/JCVM

The TSF shall provide the following **list of additional SFP capabilities: none**

5.1.1.1.6 FDP_IFF.1.5/JCVM

The TSF shall explicitly authorise an information flow based on additional rules: **none.**

5.1.1.1.7 FDP_IFF.1.6/JCVM

The TSF shall explicitly deny an information flow based on specific rules: **none.**

Note: the storage of temporary *JCRE*-owned objects' references is runtime-enforced ([JCRE22], §6.2.8.1-3).

Note that this policy essentially applies to the execution of bytecode. Native methods, the JCRE itself and possibly some API methods can be granted specific rights or limitations through the FDP_IFF.1.3/JCVM to FDP_IFF.1.6/JCVM elements. The way the virtual machine manages the transfer of values on the stack and local variables (returned values, uncaught exceptions) from and to internal registers is implementation-dependent. For the TOE access to referenced objects strictly complies with the firewall rules specified in [JCRE22] §6.2.8.1-3. To ensure this each object has access flags (set at object creation time) which are checked at each runtime access against system status variables (e.g. active group context, current object owner) to allow or to deny access. For instance, a returned reference, depending on the implementation of the stack frame, may transit trough an internal register prior to being pushed on the stack of the invoker. The **areturn** bytecode would cause more than one *OP.PUT* operation under this scheme.

FDP_RIP.1/OBJECTS Subset residual information protection

5.1.1.1.8 FDP_RIP.1.1/OBJECTS

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

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

FMT_SMF.1 Security management functions

5.1.1.1.9 FMT_SMF.1.1/JCRE

Changed by [CCFI_065]

The TSF shall be capable of performing the following security management functions: **modification of the list of registered applets' AID, modification of the active context and modification of the SELECTed applet context security attributes.**

FMT_MSA.2/JCRE Secure security attributes

5.1.1.1.10 FMT_MSA.2.1/JCRE

The TSF shall ensure that only secure values are accepted for security attributes.

Note: For instance, secure values conform to the following rules:

- The Context attribute of a **.JAVAOBJECT*[8] must correspond to that of an installed *applet* or be "JCRE".
- An *O.JAVAOBJECT* whose Sharing attribute is a *JCRE entry point* or a global array necessarily has "JCRE" as the value for its Context security attribute.
- An *O.JAVAOBJECT* whose Sharing attribute value is a global array necessarily has "array of primitive Java Card System type" as a JavaCardClass security attribute's value.
- Any *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.

Note: The above 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.

FMT_MSA.3/FIREWALL Static attribute initialisation

5.1.1.1.11 FMT_MSA.3.1/FIREWALL

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

Note: Objects' security attributes of the access control policy are created and initialized at the creation of the object or the subject. Afterwards, these attributes are no longer mutable (FMT_MSA.1/JCRE). At the creation of an object (*OP.CREATE*), the newly created object, assuming that the operation is permitted by the SFP, gets its Lifetime and Sharing attributes from the parameters of the operation; on the contrary, its Context attribute has a default value, which is its creator's Context attribute and AID respectively ([JCRE22], §6.1.2). There is one default value for the *SELECTed applet Context* that is the *default applet identifier's Context*, and one default value for the *active context*, that is "*JCRE*".

Note: There is no security attribute attached to subjects or information for this information flow policy. However, this is the JCRE who controls the currently active context. Moreover, the knowledge of which reference corresponds to a temporary entry point object or a global array and which does not is solely available to the JCRE (and the virtual machine).

5.1.1.1.12 FMT_MSA.3.2/FIREWALL

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

[8] Either subject or object.

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

FMT_SMR.1/JCRE Security roles

5.1.1.1.13 FMT_SMR.1.1/JCRE

The TSF shall maintain the roles: ***the JCRE.***

Note: the actual set of roles defined in the ST depends on the configuration.

5.1.1.1.14 FMT_SMR.1.2/JCRE

The TSF shall be able to associate users with roles.

FPT_SEP.1 TSF domain separation

5.1.1.1.15 FPT_SEP.1.1

The TSF shall maintain a ***security domain*** for its own execution that protects it from interference and tampering by untrusted subjects.

5.1.1.1.16 FPT_SEP.1.2

The TSF shall enforce separation between the security domains of subjects in the TSC.

Note: By security domain it is intended "execution context" which should not be confused with other meanings of "security domains".

5.1.1.2 Application Programming Interface

The following SFRs are related to the Java Card API.

FCS_CKM.1 Cryptographic key generation

5.1.1.2.1 FCS_CKM.1.1

The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm**: <u>3-DES -, RSA-CRT- and AES- Key Generator</u>** and specified cryptographic key sizes **<u>(112,168 bit), (1024 up to 2048 bit) and (128, 192, 256 bit)</u>** that meet the following **<u>list of standards: [JCAPI22]</u>**.

Note: The key containers are generated and diversified in accordance with [JCAPI22] specification. The actual keys are instantiated by the applet.

FCS_CKM.2 Cryptographic key distribution

5.1.1.2.2 FCS_CKM.2.1

The TSF shall distribute cryptographic KEYS in accordance with a specified cryptographic KEY distribution method

<u>DESKey.setKey()/all set-methods of class RSAPrivateCrtKey and RSAPublicKey/AESKey.setKey()</u>

that meets the following **<u>list of standards: [JCAPI22].</u>**

Note: This component is instantiated according [JCAPI22].

FCS_CKM.3 Cryptographic key access

5.1.1.2.3 FCS_CKM.3.1

The TSF shall perform **KEY access to the 3-DES/RSA/AES KEYs** in accordance with a specified cryptographic KEY access method

- **DESKey.getKey()/AESKey.getKey()**
- **All get-methods of class RSAPrivateCrtKey and RSAPublicKey**
- **All methods of class Key except clearKey()**

that meets the following **list of standards: [JCAPI22].**

Note: This component is instantiated according [JCAPI22].

FCS_CKM.4 Cryptographic key destruction

5.1.1.2.4 FCS_CKM.4.1

The TSF shall destroy cryptographic keys in accordance with a specified cryptographic KEY destruction method ***Key*.clearKey() and overwriting the keys with zeros** that meets the following **list of standards: [JCAPI22].**

Note: The keys are reset in accordance with [JCAPI22] in class Key with the method clearKey(). Any access to a cleared key attempting to use it for ciphering or signing shall throw an exception.

Note: This component is instantiated according [JCAPI22].

FCS_COP.1 Cryptographic operation

5.1.1.2.5 FCS_COP.1.1

The TSF shall perform **encryption/decryption and sign/verify** in accordance with a specified **cryptographic algorithm 3-DES in CBC/ ECB mode, RSA and AES in CBC/ECB mode with block length 128 bit** and cryptographic KEY sizes **(112,168 bit), (1024 up to 2048 bit), (128, 192, 256 bit)** that meet the following **list of standards: DES: [ISO9797] and RSA: [PKCS1].**

Note: This component is instantiated according to ([JCAPI22].

FDP_RIP.1/APDU Subset residual information protection

5.1.1.2.6 FDP_RIP.1.1/APDU

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

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

FDP_RIP.1/bArray Subset residual information protection

5.1.1.2.7 FDP_RIP.1.1/bArray

The TSF shall ensure that any previous information content of a resource is made unavailable upon the ***de-allocation of the resource from*** the following object: ***the bArray object.***

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

FDP_RIP.1/ABORT Subset residual information protection

5.1.1.2.8 FDP_RIP.1.1/ABORT

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

Note: The events that provoke the de-allocation of the previously mentioned references are described in [JCRE22], §7.6.3.

FDP_RIP.1/KEYS Subset residual information protection

5.1.1.2.9 FDP_RIP.1.1/KEYS

The TSF shall ensure that any previous information content of a resource is made unavailable upon the ***de-allocation of the resource from*** the following objects***: the cryptographic buffer (D.CRYPTO).***

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 [JCAPI22].

FDP_ROL.1/FIREWALL Basic rollback

5.1.1.2.10 FDP_ROL.1.1/FIREWALL

The TSF shall enforce ***the FIREWALL access control SFP and the JCVM information flow control SFP*** to permit the rollback of ***OP.JAVA, OP.CREATE on O.JAVAOBJECTs.***

REFINEMENT The basic rollback mechanism is realised by the SCP in the Memory Management, Cache and Transaction Layer but starts in the JCAPI.

5.1.1.2.11 FDP_ROL.1.2/FIREWALL

The TSF shall permit operations to be rolled back within the scope of a ***select(), deselect(), process() or install()*** call, notwithstanding the restrictions given in [JCRE22], §7.7, within the bounds of the Commit Capacity ([JCRE22], §7.8), and those described in [JCAPI22]

REFINEMENT The basic rollback mechanism is realised by the SCP in the Memory Management, Cache and Transaction Layer but starts in the JCAPI.

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 relies on the transaction mechanism offered by the platform. Some operations of the API are not conditionally updated, as documented in [JCAPI22] (see for instance, PIN-blocking, PIN-checking, update of Transient objects).

Note: The loading and linking of applet packages (the installation or registration is covered by FDP_ROL.1.1/FIREWALL) is subject to some kind of rollback mechanism (see FPT_RCV.3.1/Installer), described in [JCRE22], §11.2.

5.1.1.3 Card Security Management

The following SFRs are related to the security requirements at the level of the whole card, in contrast to the previous ones, that are somewhat restricted to the TOE alone. For instance, a potential security violation detected by the virtual machine may require a reaction that does not only concern the virtual machine, such as blocking the card (or request the appropriate security module with the power to block the card to perform the operation).

FAU_ARP.1/JCS Security alarms

5.1.1.3.1 FAU_ARP.1.1/JCS

The TSF shall take ***throw an exception, lock the card session or reinitialize the Java Card System and its data*** **<u>and other actions: none</u>** upon detection of a potential security violation.

REFINEMENT Potential security violation is refined to 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
- Abortion of a transaction in an unexpected context (see (**`abortTransaction()`**,[JCAPI22] and ([JCRE22], §7.6.2)
- Violation of the Firewall or JCVM SFPs
- Unavailability of resources
- Array overflow
- Other runtime errors related to *applet*'s failure (like uncaught exceptions)

Note: The thrown exceptions and their related events are described in [JCRE22], [JCAPI22] and [JCVM22].

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. For the TOE in this ST bytecode verification is performed off-card.

Note: Depending on the context of use and the required security level, there are cases where the card manager and the other part of the TOE must work in cooperation to detect and appropriately react in case of potential security violation. This behaviour is described in this component. It details 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).

Note: The "locking of the card session" does 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 occurs when "clean" re-initialization is otherwise impossible.

The locking may be implemented at the level of the *Java Card System* as a denial of service (through some systematic "fatal error" message or return value) that lasts up to the next "RESET" event, without affecting other components of the card (such as the card manager).

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

FDP_SDI.2 Stored data integrity monitoring and action

5.1.1.3.2 FDP_SDI.2.1

The TSF shall monitor user data stored within the TSC for **integrity errors** on all objects, based on the following attributes**: checksum integrity of cryptographic keys, PIN values and their associated security attributes.**

5.1.1.3.3 FDP_SDI.2.2

Upon detection of a data integrity error, the TSF shall **bring the card into a secure state**.

Note: Although no such requirement is mandatory in the specification, an exception is 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 are considered with particular attention as they play a key role in the overall security.

Note: Integrity errors in the code of the native applications and Java Card technology-based applications ("Java Card applications") are monitored.

For integrity sensitive application~~,~~ their data shall be monitored (D.APP_I_DATA): applications may need to protect information against unexpected modifications~~,~~ and explicitly control whether a piece of information has been changed between two accesses. For example, maintaining the integrity of an electronic purse's balance is extremely important because this value represents real money. Its modification must be controlled, for illegal ones would denote an important failure of the payment system.

A dedicated library could be implemented and made available to developers to achieve better security for specific objects, following the same pattern that already exists in cryptographic APIs, for instance.

FPT_RVM.1 Non-bypass ability of the TSP

5.1.1.3.4 FPT_RVM.1.1

The TSF shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

Note: Execution of native code is not within the TSC. Nevertheless, access to native methods from the Java Card System is subject to TSF control, as there is no difference in the interface or the invocation mechanism between native and interpreted methods.

FPT_TDC.1 Inter-TSF basic TSF data consistency

5.1.1.3.5 FPT_TDC.1.1

The TSF shall provide the capability to consistently interpret ***the CAP files*** (shared between the card manager and other parts of the TOE), ***the bytecode and its data arguments*** (shared with applets and API packages), when shared between the TSF and another trusted IT product.

Note: Concerning the interpretation of data between the Java Card System and the Smart Card platform, the TOE is developed consistently with the *SCP* functions, namely concerning memory management, I/O functions, cryptographic functions, and so on.

5.1.1.3.6 FPT_TDC.1.2

The TSF shall use the following rules when interpreting the TSF data from another trusted IT product:

- ***The [JCVM22] specification;***
- ***Reference export files;***
- ***The ISO 7816-6 rules;***
- ***The EMV specification***.

FPT_FLS.1/JCS Failure with preservation of secure state

5.1.1.3.7 FPT_FLS.1.1/JCS

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

Note: The JCRE Context is the Current context when the VM begins running after a card reset ([JCRE22], §6.2.3). Behaviour of the TOE on power loss and reset is described in [JCRE22], §3.6, and §7.1.

FPR_UNO.1 Unobservability

5.1.1.3.8 FPR_UNO.1.1

The TSF shall ensure that **S.PACKAGE** is unable to observe **cryptographic operations** on **3-DES/RSA/AES Keys and PIN objects and comparisons performed on PINs owned** by **another S.PACKAGE**.

Note: Although it is not required in [JCRE22] specifications, the non-observability of operations on sensitive information such as keys appears as impossible to circumvent in the smart card world. The precise list of operations and objects is left unspecified, but should at least concern secret keys and PIN codes when they exists on the card, as well as the cryptographic operations and comparisons performed on them.

FPT_TST.1 TSF testing

5.1.1.3.9 FPT_TST.1.1

The TSF shall run a suite of self tests ***during initial start-up (at each power on)*** to demonstrate the correct operation of the TSF.

REFINEMENT The TSF testing is realised by the SCP in the Memory Management and Service Layer.

Note: TSF-testing is not mandatory in [JCRE22], but appears in most of security requirements documents for masked applications. Testing could also occur randomly

5.1.1.3.10 FPT_TST.1.2

The TSF shall provide authorised users with the capability to verify the integrity of the TSF data.

REFINEMENT The TSF testing is realised by the SCP in the Memory Management and Service Layer.

5.1.1.3.11 FPT_TST.1.3

The TSF shall provide authorised users with the capability to verify the integrity of stored TSF executable code.

REFINEMENT The TSF testing is realised by the SCP in the Memory Management and Service Layer.

5.1.1.4 AID Management

The following SFRs are related to AID management.

FMT_MTD.1/JCRE Management of TSF data

5.1.1.4.1 FMT_MTD.1.1/JCRE

The TSF shall restrict the ability to ***modify*** *the* ***list of registered applets' AID*** *to* ***the JCRE*** **and other authorized identified roles: none.**

Note: The *installer* and the *JCRE* manage some other TSF data such as the *applet* life cycle or *CAP file*s. Objects in the Java programming language may also try to query *AIDs* of installed *applet*s through the **lookupAID(...)** API method.

Note: The *installer, applet deletion manager* or even the card manager may be granted the right to modify the list of registered applets' *AIDs* in specific implementations (possibly needed for installation and deletion; see #.DELETION and #.INSTALL). For this TOE the Issuer Security Domain (Card Manager Applet) is implemented according to the GlobalPlatform specification 2.1.1 The Issuer Security Domain is the installer and deletion manager simultaneously and belongs to the JCRE (has JCRE rights). The list of registered applets' AIDs is part of the Registry which belongs to the JCRE too. Upon installation/deletion the Issuer Security Domain adds/removes the corresponding entry to/from the list of registered applets' AIDs via the ContentManager object which is a JCRE Entry Point Object. However, the Issuer Security Domain may also change its AID at the receipt of a STORE DATA command.

FMT_MTD.3 Secure TSF data

5.1.1.4.2 FMT_MTD.3.1

The TSF shall ensure that only secure values are accepted for TSF data.

FIA_ATD.1/AID User attribute definition

5.1.1.4.3 FIA_ATD.1.1/AID

The TSF shall maintain the following list of security attributes belonging to individual users: ***the AID and version number of each package, the AID of each registered applet, and whether a registered applet is currently selected for execution (***[JCVM22]***, §6.5).***

FIA_UID.2/AID User identification before any action

5.1.1.4.4 FIA_UID.2.1/AID

The TSF shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user.

Note: By users here it must be understood the ones associated to the packages (or applets) which act as subjects of policies. In the Java Card System, every action is always performed by an identified user interpreted here as the currently selected applet or the package that is the subject's owner. Means of identification are provided during the loading procedure of the package and the registration of applet instances.

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

FIA_USB.1 User-subject binding

5.1.1.4.5 FIA_USB.1.1

Changed by [CCFI_137] to:

The TSF shall associate the following user security attributes with subjects acting on behalf of that user: **Package AID, or "JCRE".**

Note: For S.PACKAGEs, the Context security attribute plays the role of the appropriate user security attribute; see above.

5.1.1.4.6 FIA_USB.1.2

Changed by [CCFI_137] to:

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

5.1.1.4.7 FIA_USB.1.3

Changed by [CCFI_137] to:

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.**

5.1.2 InstG Security Functional Requirements

This group bulks the SFRs related to the installation of the *applet*s, which addresses security aspects outside the runtime. The idea here is that installation of *applet*s is a critical phase, which lies partially out of the boundaries of the *firewall*, and therefore has to be deserved specific treatment. In the Common Criteria model, loading a *package* or installing an *applet* was considered as being an importation of user data (that is, user application‘s data) with its security attributes (such as the parameters of the *applet* used in the firewall rules).

See also FIA_ATD.1, FIA_USB.1, FMT_MTD.1, FMT_SMR.1 for various information about *applet* installation.

FDP_ITC.2 Import of user data with security attributes

5.1.2.1 FDP_ITC.2.1/Installer

The TSF shall enforce the ***FIREWALL access control SFP*** when importing user data, controlled under the SFP, from outside of the TSC.

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

5.1.2.2 FDP_ITC.2.2/Installer

The TSF shall use the security attributes associated with the imported user data.

5.1.2.3 FDP_ITC.2.3/Installer

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

Note: The format of the CAP file is precisely defined in Sun's specification ([JCVM22]); it contains the user data (like applet's code and data) and the security attributes altogether. Therefore there is no association to be carried out elsewhere.

5.1.2.4 FDP_ITC.2.4/Installer

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

Note: Each package contains a package Version attribute, which is a pair of major and minor version numbers ([JCVM22], §4.5). With the AID, it describes the package defined in the CAP file. When an export file is used during preparation of a CAP file, the versions numbers and AIDs indicated in the export file are recorded in the CAP files ([JCVM22], §4.5.2): the dependent packages Versions and AIDs attributes allow the retriev~~a~~l of these identifications. Implementation-dependent checks may occur on a case-by-case basis to indicate that *package* files are binary compatibles. These checks for binary compatibility are executed for the TOE during processing the import component. The following comparison is done: The package version number contained in the import component is compared with the version number of the oncard implementation of this package (stored in the oncard registry). However, *package* files do have "*package* Version Numbers" ([JCVM22]) used to indicate binary compatibility or incompatibility between successive implementations of a *package*, which obviously directly concern this requirement.

5.1.2.5 FDP_ITC.2.5/Installer

The TSF shall enforce the following rule when importing user data controlled under the SFP from outside the TSC:

A *package* may depend on (import or use data from) other *package*s already installed. This dependency is explicitly stated in the loaded *package* in the form of a list of *package AIDs*. The loading is allowed only if, for each dependent *package*, its *AID* attribute is equal to a resident package *AID* attribute, the major (minor) Version attribute associated to the former is equal (less than or equal) to the major (minor) Version attribute associated to the latter ([JCVM22],§4.5.2). The intent of this rule is to ensure the binary compatibility of the *package* with those already on the card ([JCVM22], §4.4).

Note: The installation (the invocation of an *applet*'s install method by the *installer*) is implementation dependent ([JCRE22], §11). For this TOE the invocation of an applet's install method is its install_method offset within the CAP Method Component of a package, owned by the needed Applet class. Furthermore for each Applet package the package AID is entered in the registry.

Upon the APDU INSTALL [for install] Command (which contains the package AID and the Applet class AID) the ContentManager searches for the associated install_method_offset. If found, then the install() method will be invoked.

Note: Other rules governing the installation of an *applet*, that is, its registration to make it SELECTable by giving it a unique *AID*, are also implementation dependent (see, for example, ([JCRE22], §11.1.2). For this TOE upon APDU INSTALL [for install] (which contains the applet instance AID) the ContentManager checks if this applet instance AID is already in use by an other applet or package and in this case rejects the command.

FMT_SMR.1 Security roles

5.1.2.6 FMT_SMR.1.1/Installer

The TSF shall maintain the roles: *S.CRD*[9].

Note: the actual set of roles defined in the ST depends on the configuration.

5.1.2.7 FMT_SMR.1.2/Installer

The TSF shall be able to associate users with roles.

FPT_FLS.1 Failure with preservation of secure state

5.1.2.8 FPT_FLS.1.1/Installer

The TSF shall preserve a secure state when the following types of failures occur: ***the installer fails to load/install a package/applet as described in ([JCRE22] §11.1.5.***

Note: The TOE does not provide additional feedback information to the card manager in case of potential security violations (see FAU_ARP.1).

FPT_RCV.3 Automated recovery without undue loss

5.1.2.9 FPT_RCV.3.1/Installer

When automated recovery from a failure or service discontinuity is not possible, the TSF shall enter a maintenance mode where the ability to return the TOE to a secure state is provided.

Note: This element is not within the scope of the Java Card specification, which only mandates the behavior of the Java Card System in good working order. The following is an excerpt from [CC1]:

> *In this maintenance mode normal operation might be impossible or severely restricted, as otherwise insecure situations might occur. Typically, only authorized users should be allowed access to this mode but the real details of who can access this mode is a function of class FMT Security management. If FMT does not put any controls on who can access this mode, then it may be acceptable to allow any user to restore the system if the TOE enters such a state. However, in practice, this is probably not desirable as the user restoring the system has an opportunity to configure the TOE in such a way as to violate the TSP.*

5.1.2.10 FPT_RCV.3.2/Installer

For **reset, insufficient EEPROM, failure in cryptographic safeguarding, package references (versions) mismatching,** the TSF shall ensure the return of the TOE to a secure state using automated procedures.

Note: Should the *installer* fail during loading/installation of a *package/applet*, it reverts to a "consistent and secure state". The *JCRE* has some clean up duties as well; see ([JCRE22], §7.6.3 for possible scenarios. Precise behavior is left to implementers.

Note: In this ST the configuration includes the *applet deletion manager* (and the associated group ADELG). This component includes among the listed failures that of the deletion of a *package/applet*. See ([JCRE22], 11.3.4) for possible scenarios.

[9] The term installer of the PP is replaced here by S.CRD to make the ST more consistent.

Other events such as the unexpected tearing of the card, power loss, and so on are partially handled by the hardware platform (see the SCPG group) and the Java Card System "that clear transient objects" and transactional features. See FPT_FLS.1.1/JCS, FDP_RIP.1.1/TRANSIENT, FDP_RIP/ABORT and FDP_ROL.1.

5.1.2.11 FPT_RCV.3.3/Installer

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

> Note: The *SCP* ensures the atomicity of updates for fields and objects (see the SCPG group), and a power-failure during a transaction or the normal runtime does not create the loss of otherwise-permanent data, in the sense that memory on a smart card is essentially persistent with this respect (EEPROM). Data stored on the RAM and subject to such failure is intended to have a limited lifetime anyway (runtime data on the stack, transient objects' contents). According to this, the loss of data within the TSC should be limited to the same restrictions of the transaction mechanism.

5.1.2.12 FPT_RCV.3.4/Installer

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

FRU_RSA.1 Maximum quotas

5.1.2.13 FRU_RSA.1.1/Installer

The TSF shall enforce maximum quotas of the following resources: **imported packages** and **declared classes**, **methods and fields** that **packages** can use **simultaneously**.

> Note: A package may import at most 128 packages and declare at most 255 classes and interfaces. A *class* can implement a maximum of 128 public or protected instance methods and a maximum of 128 instance methods with *package* visibility. These limits include inherited methods. A *class* instance can contain a maximum of 255 fields, where an **int** data type is counted as occupying two fields ([JCVM22], §2.2.4.2).

5.1.3 ADELG Security Functional Requirements

This group bulks the SFRs related to the deletion of *applet*s and/or *packages*, enforcing the *applet deletion manager* **(ADEL) policy** on security aspects outside the runtime. The idea here is that deletion is a critical phase and therefore requires specific treatment.

Applet Deletion Manager Policy

FDP_ACC.2: Complete access control

5.1.3.1 FDP_ACC.2.1/ADEL

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

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

S.ADEL	The *applet deletion manager*. It may be an *applet* ([JCRE22], §11), but its role asks anyway for a specific treatment from the security viewpoint. *This subject is unique.*
O.CODE_PKG	The code of a *package*, including all linking information. On the Java Card platform, a package is the installation unit.
O.APPLET	Any installed *applet*, its code and data.
O.JAVAOBJECT	Java class instance or array.

Operations (prefixed with "*OP*") of this policy are described in the following table.

Operation	**Description**
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed *applet* and its objects, either logically or physically.
OP.DELETE_PCKG(O.CODE_PKG,...)	Delete a *package*, either logically or physically
OP.DELETE_PCKG_APPLET(O.CODE_PKG,...)	Delete a *package* and its installed *applet*s, either logically or physically.

Table 4 Operations of the Applet Deletion Manager Policy

5.1.3.2 FDP_ACC.2.2/ADEL

The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

FDP_ACF.1 Security attribute based access control

5.1.3.3 FDP_ACF.1.1/ADEL

Changed by [CCFI_103] to:

The TSF shall enforce the ADEL access control SFP to objects based on the following: O.CODE_PKG, O.APPLET and O.JAVAOBJECT and on the relevant security attributes:
(1) the security attributes of the covered subjects and objects,
(2) the list of AIDs of the applet instances registered on the card,
(3) the attribute ResidentPackages, which journals the list of AIDs of the packages already loaded on the card, and
(4) the attribute ActiveApplets, which is a list of the active applets' AIDs.

The following table presents some of the security attributes associated to the subjects/objects under control of the policy. However, they are mostly implementation independent.

Subject/Object	Attributes
O.CODE_PKG	*package*'s *AID*, dependent packages' *AID*s, Static References
O.APPLET	Selection state
O.JAVAOBJECT	Owner, Remote

Table 5 Security attributes associated to the subjects/objects under control of the ADEL access control policy

The package's *AID* identifies the package defined in the CAP file.

When an export file is used during preparation of a CAP file, the version numbers and *AIDs* indicated in the export file are recorded in the CAP files ([JCVM22], §4.5.2): the dependent packages AIDs attribute allows the retrieval of those identifications.

Static fields of a package may contain references to objects. The Static References attribute records those references.

An applet instance can be in two different selection states: selected or deselected. If the applet is selected (in some logical channel), then in turn it could either be *currently selected* or just *active*. At any time there could be up to four active applet instances, but only one currently selected. This latter is the one that is processing the current command ([JCRE22], §4).

The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the package).

An object is said to be a Remote *if* it is an instance of a class that directly or indirectly implements the interface **`java.rmi.Remote`**.

Finally, there are needed security attributes that are not attached to any object or subject of the TSP: (1) the ResidentPackages Versions (or Resident Image, [JCVM22]§4.5) and *AID*s. They describe the packages that are already on the card, (2) the list of registered applet instances and (3) the ActiveApplets security attribute. They are all attributes internal to the VM, that is, not attached to any specific object or subject of the SPM. These attributes are TSF data that play a role in the SPM.

5.1.3.4 FDP_ACF.1.2/ADEL

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

The subject of this policy is *S.ADEL*.

Some basic common specifications are required in order to allow Java Card *applet*s and *package*s to be deleted without knowing the implementation details of a particular deletion manager. In particular, this policy introduces a notion of **reachability**, which provides a general means to describe objects that are referenced from a certain *applet* instance or *package*.

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

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

R.JAVA.14 ([JCRE22], §11.3.4.1, **Applet Instance Deletion***)*. The *S.ADEL* may perform *OP.DELETE_APPLET* upon an *O.APPLET* only if, (1) *S.ADEL* is currently selected, (2) *O.APPLET* is deselected and (3) there is no *O.JAVAOBJECT* owned by *O.APPLET* such that either *O.JAVAOBJECT* is reachable from an applet instance distinct from *O.APPLET*, or *O.JAVAOBJECT* is reachable from a package P, or ([JCRE22], §8.5) *O.JAVAOBJECT* is remote reachable.

R.JAVA.15 ([JCRE22],§11.3.4.1, **Multiple Applet Instance Deletion***)*. The *S.ADEL* may perform *OP.DELETE_APPLET* upon several *O.APPLET* only if, (1) *S.ADEL* is currently selected, (2) every *O.APPLET* being deleted is deselected and (3) there is no *O.JAVAOBJECT* owned by any of the *O.APPLET* being deleted such that either *O.JAVAOBJECT* is reachable from an applet instance distinct from any of those *O.APPLET*, or *O.JAVAOBJECT* is reachable from a package P, or ([JCRE22], §8.5) *O.JAVAOBJECT* is remote reachable.

R.JAVA.16 ([JCRE22], §11.3.4.2, **Applet/Library Package Deletion***)*. The *S.ADEL* may perform *OP.DELETE_PCKG* upon an *O.CODE_PCKG* only if, (1) *S.ADEL* is currently selected, (2) no reachable *O.JAVAOBJECT*, from a package distinct from *O.CODE_PCKG* that is an instance of a class that belongs to *O.CODE_PCKG* exists on the card and (3) there is no package loaded on the card that depends on *O.CODE_PCKG.*

R.JAVA.17 ([JCRE22], §11.3.4.3, Applet **Package and Contained Instances Deletion***)*. The *S.ADEL* may perform *OP.DELETE_PCKG_APPLET* upon an *O.CODE_PCKG* only if, (1) *S.ADEL* is currently selected, (2) no reachable *O.JAVAOBJECT*, from a package distinct from *O.CODE_PCKG*, which is an instance of a class that belongs to *O.CODE_PCKG* exists on the card, (3) there is no package loaded on the card that depends on *O.CODE_PCKG* and (4) for every *O.APPLET* of those being deleted it holds that: (i) *O.APPLET* is deselected and (ii) there is no *O.JAVAOBJECT* owned by *O.APPLET* such that either *O.JAVAOBJECT* is reachable from an applet instance not being deleted, or *O.JAVAOBJECT* is reachable from a package not being deleted, or ([JCRE22],§8.5) *O.JAVAOBJECT* is remote reachable.

5.1.3.5 FDP_ACF.1.3/ADEL

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

Note: However, the *S.ADEL* are granted privileges ([JCRE22], §11.3.5) to bypass the preceding policies. For instance, the logical deletion of an *applet* renders it un-selectable; this has implications on the management of the associated TSF data (see note of FMT_MTD.1.1/JCRE).

5.1.3.6 FDP_ACF.1.4/ADEL

The TSF shall explicitly deny access of any subject but the *S.ADEL* to *O.CODE_PKG* or *O.APPLET* for the purpose of deleting it from the card.

FMT_MSA.1 Management of security attributes

5.1.3.7 FMT_MSA.1.1/ADEL

The TSF shall enforce the ADEL *access control SFP* to restrict the ability **to modify the ActiveApplets** security attribute to **the JCRE (*S.JCRE*).**

Note: The modification of the ActiveApplets security attribute are performed in accordance with the rules given in [JCRE22], §4.

FMT_MSA.3 Static attribute initialization

5.1.3.8 FMT_MSA.3.1/ADEL

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

5.1.3.9 FMT_MSA.3.2/ADEL

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

FMT_SMR.1 Security roles

5.1.3.10 FMT_SMR.1.1/ADEL

The TSF shall maintain the roles: ***S.ADEL***[10].

Note: the actual set of roles defined in the ST depends on the configuration.

5.1.3.11 FMT_SMR.1.2/ADEL

The TSF shall be able to associate users with roles.

Additional Deletion Requirements

FDP_RIP.1 Subset residual information protection

5.1.3.12 FDP_RIP.1.1/ADEL

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

Note: Deleted freed resources (both code and data) are reused, depending on the way they were deleted (logically or physically). Requirements on *de-allocation* during *applet/package* deletion are described in [JCRE22], §11.3.4.1, §11.3.4.2 and §11.3.4.3.

Note: There is no conflict with FDP_ROL.1.1 requirements appearing in the document as of the bounds on the rollback: the deletion operation is out of the scope of the rollback (FDP_ROL.1.1/FIREWALL).

FPT_FLS.1 Failure with preservation of secure state

[10] The applet deletion manager of the PP is replaced here by S.ADEL to make the ST more consistent.

5.1.3.13 FPT_FLS.1.1/ADEL

The TSF shall preserve a secure state when the following types of failures occur: ***the*** applet deletion manager ***fails to delete a package/applet as described in [JCRE22], §11.3.4.***

Note: The TOE does not provide additional feedback information to the card manager in case of a potential security violation (see FAU_ARP.1).

5.1.4 RMIG Security Functional Requirements

This group is mainly devoted to specifying the policies that control the access to remote objects and the flow of information that takes place when the RMI service is used. There are specific control rules concerning the access to remote objects. The rules relate mainly to the lifetime of their corresponding remote references. Information concerning remote object references can be sent out of the card only if the corresponding remote object has been designated as exportable. Array parameters of remote method invocations are required to be allocated on the card as global arrays, the storage of references to those arrays must then be restricted as well.

JCRMI Policy

The *JCRMI* policy embodies both an access control and an information flow control policy.

FDP_ACC.2: Complete access control

5.1.4.1 FDP_ACC.2.1/JCRMI

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

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

S.CAD The *CAD*. In the scope of this policy it represents the actor that requests, by issuing commands to the card, for RMI services.

S.JCRE The *JCRE* is responsible on behalf of the card issuer of the bytecode execution and runtime environment functionalities. In the context of this security policy, the *JCRE* is in charge of the execution of the commands provided to (1) obtain the initial remote reference of an applet instance and (2) perform Remote Method Invocation.

O.APPLET Any installed *applet*, its code and data.

O.REMOTE_OBJ A remote object is an instance of a class that implements one (or more) remote interfaces. A remote interface is one that extends, directly or indirectly, the interface **java.rmi.Remote** ([JCAPI22]).

O.ROR A remote object reference. It provides information concerning: (i) the identification of a remote object and (ii) the Implementation class of the object or the interfaces implemented by the class of the object. This is the object's information to which the CAD can access.

O.REMOTE_MTHD A method of a remote interface.

O.RMI_SERVICE These are instances of the class **javacardx.rmi.RMIService**. They are the objects that actually process the RMI services.

Operations (prefixed with "*OP*") of this policy are described in the following table.

Operation	Description
OP.GET_ROR(O.APPLET,…)	Retrieves the initial remote object reference of a RMI based applet. This reference is the seed which the CAD client application needs to begin remote method invocations
OP.INVOKE(O.RMI_SERVICE,…)	Requests a remote method invocation on the remote object.

5.1.4.2 FDP_ACC.2.2/JCRMI

The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

FDP_ACF.1 Security attribute based access control

5.1.4.3 FDP_ACF.1.1/JCRMI

The TSF shall enforce the JCRMI access control SFP to objects based on: ***(1) the security attributes of the covered subjects and objects, (2) the list of AIDs of the applet instances registered on the card and (3) the attribute ActiveApplets, which is a list of the active applets' AIDs.***

The following table presents the security attributes associated to the objects under control of the policy.

Object	Attributes
O.APPLET	*Package*'s *AID* or *none*
O.REMOTE_OBJ	Owner, class, Identifier, Exported
O.REMOTE_MTHD	Identifier
O.RMI_SERVICE	Owner, Returned References

The package's *AID* identifies the package defined in the CAP file.

An applet instance can be in two different selection states: selected or deselected. If the applet is selected (in some logical channel), then in turn it could either be *currently selected* or just *active*. At any time there could be up to four active applet instances, but only one currently selected. This latter is the one that is processing the current command ([JCRE22], §4).

The **owner** of a remote object is the applet instance that created the object. The **class** attribute identifies the implementation class of the remote object. The **remote object**

Identifier is a number that uniquely identifies a remote object. The attribute **Exported** indicates whether the remote object is exportable or not.

A **remote method Identifier** is a number that uniquely identifies a remote method within a certain remote class.

The **owner** of an *O.RMI_SERVICE* is the applet instance that created the object. The attribute **Returned References** lists the remote object references that have been sent to the CAD during the applet selection session. This attribute is implementation dependent.

Finally, there are some security attributes that are not attached to any object or subject of the TSP: (1) the list of registered applet instances and (2) the ActiveApplets security attribute. They are all attributes internal to the VM that is, not attached to any specific object or subject of the SPM. These attributes are TSF data that play a role in the SPM.

5.1.4.4 FDP_ACF.1.2/JCRMI

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

R.JAVA.6 The *S.CAD* may perform *OP.GET_ROR* upon an *O.APPLET* only if *O.APPLET* is the *currently selected applet*, and there exists an *O.RMI_SERVICE* with a registered initial reference to an *O.REMOTE_OBJ* that is owned by *O.APPLET*.

R.JAVA.7 The *S.JCRE* may perform *OP.INVOKE* upon *O.RMI_SERVICE*, *O.ROR* and *O.REMOTE_MTHD*, only if, *O.ROR* is valid (as defined in [JCRE22], §8.5) and belongs to the value of the attribute Returned References of *O.RMI_SERVICE*, and the attribute Identifier of *O.REMOTE_MTHD* matches one of the remote methods in the class, indicated by the security attribute class, of the *O.REMOTE_OBJECT* to which *O.ROR* makes reference.

Note: The validity of a remote object reference is specified as a lifetime characterization. The security attributes involved in the rules that determine what a valid remote object reference is are the attribute Returned References of the O.RMI_SERVICE and the attribute ActiveApplets (see FMT_REV.1.1/JCRMI and FMT_REV.1.2/JCRMI).

Note: The precise mechanism by which a remote method is invoked on a remote object is defined in detail in ([JCRE22], §8.5.2 and [JCAPI22]).

5.1.4.5 FDP_ACF.1.3/JCRMI

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

5.1.4.6 FDP_ACF.1.4/JCRMI

The TSF shall explicitly deny access of ***any subject but S.JCRE*** to ***O.remote_obj and O.remote_mthd*** for the purpose of performing a remote method invocation.

FDP_IFC.1 Subset information flow control

5.1.4.7 FDP_IFC.1.1/JCRMI

The TSF shall enforce the ***JCRMI information flow control SFP*** on the following subjects, information and operations.

Subjects[11] (prefixed with an "S") and information (prefixed with an "I") covered by this policy are:

Subject/Information	Description
S.JCRE	As in the Access control policy
S.CAD	As in the Access control policy
I.RORD	Remote object reference descriptors

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

> Note: Array parameters of remote method invocations are allocated on the card as global arrays objects. References to global arrays cannot be stored in *class* variables, instance variables or array components. The control of the flow of that kind of information has already been specified in FDP_IFC.1.1/JCVM.

There is a unique operation in this policy:

Operation	Description
OP.RET_RORD(S.JCRE,S.CAD,I.RORD)	Send a remote object reference descriptor to the CAD.

A remote object reference descriptor is sent from the card to the CAD either as the result of a successful applet selection command ([JCRE22], §8.4.1), and in this case it describes, if any, the initial remote object reference of the selected applet; or as the result of a remote method invocation ([JCRE22],§8.3.5.1) .

FDP_IFF.1 Simple security attributes

5.1.4.8 FDP_IFF.1.1/JCRMI

The TSF shall enforce the ***JCRMI information flow control SFP*** based on the following types of subject and information security attributes: ***the security attribute Exported of the information.***

The following table summarizes which security attribute is attributed to which subject/information.

Subject/Information	Attributes
S.JCRE	None
S.CAD	None

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

Subject/Information	Attributes
I.RORD	ExportedInfo (Boolean value)

The ExportedInfo attribute of an *I.RORD* indicates whether the *O.REMOTE_OBJ* which *I.RORD* identifies is exported or not (as indicated by the security attribute Exported of the *O.REMOTE_OBJ*).

5.1.4.9 FDP_IFF.1.2/JCRMI

The TSF shall permit an information flow between a controlled subject and controlled information through a controlled operation if the following rule holds:

An operation *OP.RET_RORD(S.JCRE, S.CAD, I.RORD)* is permitted only if the attribute ExportedInfo *I.RORD* has the value **"true"** ([JCRE22], §8.5).

5.1.4.10 FDP_IFF.1.3/JCRMI

The TSF shall enforce **additional information flow control SFP rules: none**.

5.1.4.11 FDP_IFF.1.4/JCRMI

The TSF shall provide **list of additional SFP capabilities: none.**

5.1.4.12 FDP_IFF.1.5/JCRMI

The TSF shall explicitly authorize an information flow based on the following rules: **none.**

5.1.4.13 FDP_IFF.1.6/JCRMI

The TSF shall explicitly deny an information flow based on the following rules: **An operation *OP.RET_RORD(S.JCRE, S.CAD, I.RORD)* is denied if the attribute ExportedInfo *I.RORD* has the value "false" ([JCRE22], §8.5).**

FMT_MSA.1 Management of security attributes

5.1.4.14 FMT_MSA.1.1/JCRMI

The TSF shall enforce the ***FIREWALL access control SFP and the JCVM information flow control SFP*** to restrict the ability to **modify the ActiveApplets security attribute to the JCRE (*S.JCRE*).**

Note: The modification of the ActiveApplets security attribute is performed in accordance with the rules given in [JCRE22], §4.

5.1.4.15 FMT_MSA.1.1/EXPORT

The TSF shall enforce the ***JCRMI access control SFP and the JCRMI information flow control SFP*** to restrict the ability to **modify the security attribute Exported of an O.REMOTE_OBJ to its owner.**

Note: The Exported status of a remote object can be modified by invoking its methods export() and unexport(), and only the owner of the object may perform the invocation without raising a SecurityException (javacard.framework.service.CardRemoteObject). However, even if the owner of the object may provoke the change of the security attribute value, the modification itself is performed by the JCRE.

5.1.4.16 FMT_MSA.1.1/REM_REFS

The TSF shall enforce the *JCRMI access control SFP and the JCRMI information flow control SFP* to restrict the ability **to modify the security attribute Returned References of an O.RMI_SERVICE to its owner.**

FMT_MSA.3 Static attribute initialization

5.1.4.17 FMT_MSA.3.1/JCRMI

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

Note: Remote objects' security attributes are created and initialized at the creation of the object, and except for the Exported attribute, the values of the attributes are not longer modifiable. The default value of the Exported attribute is true.

Note: There is one default value for the *SELECTed applet context* that is the *default applet identifier's context*, and one default value for the *active context*, that is "*JCRE*".

5.1.4.18 FMT_MSA.3.2/JCRMI

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

Note: The intent is to have none of the identified roles to have privileges with regards to the default values of the security attributes. Notice that creation of objects is an operation controlled by the ***FIREWALL SFP***; the latitude on the parameters of this operation is described there.

FMT_REV.1 Revocation

5.1.4.19 FMT_REV.1.1/JCRMI

The TSF shall restrict the ability to revoke the **Returned References security attribute of an O.RMI_SERVICE** to **the JCRE** <u>**other authorized identified role: none**</u>**.**

5.1.4.20 FMT_REV.1.2/JCRMI

The TSF shall enforce the rules ***that determine the lifetime of remote object references.***

Note: The rules previously mentioned are described in [JCRE22], §8.5.

FMT_SMR.1 Security roles

5.1.4.21 FMT_SMR.1.1/JCRMI

The TSF shall maintain the roles: ***applet***.

*Note: applet*s own Remote interface objects and may choose to allow or forbid their exportation, which is managed through a security attribute.

Note: the actual set of roles defined in the ST depends on the configuration.

5.1.4.22 FMT_SMR.1.2/JCRMI

The TSF shall be able to associate users with roles.

5.1.5 LCG Security Functional Requirements

The security issues introduced by *logical channels* are mainly related to the access to *SIO* objects owned by legacy *applets* as well as to the clearing of transient data which is shared by *applet* instances which are concurrently active in different *logical channels*. Accordingly, this group introduces a reformulation of the ***FIREWALL SFP*** specified in the group CoreG and a modification to a component of the security requirement for residual information protection (*FDP_RIP.1.1/TRANSIENT*).

Firewall Policy

Except for the requirements explicitly introduced in what follows, this policy includes unchanged the functional requirements specified in the ***FIREWALL*** *access control* ***SFP*** of the group CoreG.

FDP_ACC.2: Complete access control

5.1.5.1 FDP_ACC.2.1/ FIREWALL

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

Subjects (prefixed with an "*S*"), objects (prefixed with an "*O*") and operations (prefixed with "OP") are exactly the same which are covered by the ***FIREWALL*** access control SFP.

FDP_ACF.1 Security attribute based access control

See FMT_MSA.1 for more information about security attributes.

5.1.5.2 FDP_ACF.1.1/FIREWALL

The TSF shall enforce the ***FIREWALL access control SFP*** to objects based on: ***(1) the security attributes of the covered subjects and objects, (2) the currently active context, (3) the SELECTed applet Context, and (4) the attribute ActiveApplets, which is a list of the active applets' AIDs.***

The following table describes the new security attribute attached to the subjects *S.PACKAGE*

Subject	Attributes
S.PACKAGE	Selection Status

The following table describes the possible values for the new security attributes.

Name	Description
Selection Status	Multiselectable, Non-multiselectable or "None"

Name	Description
ActiveApplets	List of package's *AID*s

The Java Card platform, version 2.2, 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 applet*s. *Applet*s that belong to a same *package* are either all multiselectable or not ([JCVM22], §2.2.5). Therefore, the selection mode can be regarded as an attribute of *package*s. No selection mode is defined for a library *package*.

Support for multiple *logical channels* (with multiple selected applet instances) requires a change to the Java Card System, version 2.1.1, concept of *selected applet*. Since more than one applet instance can be selected at the same time, and one *applet* instance can be selected on different *logical channels* simultaneously, it is necessary to differentiate the state of the *applet* instances in more detail. An *applet* instance will be considered an *active applet instance* if it is currently selected in at least one logical channel, up to a maximum of four. 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. ([JCRE22], §4).

The ActiveApplets security attribute is internal to the VM, that is, not attached to any specific object or subject of the SPM. The attribute is TSF data that plays a role in the SPM.

5.1.5.3 FDP_ACF.1.2/ FIREWALL

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

The same rules of the ***FIREWALL SFP*** defined in 5.1.1.1 except for rule **R.JAVA.4**, which must be replaced by the following rule:

R.JAVA.8 ([JCRE22], §6.2.8.6) An *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 one of the following applies:

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

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

and in either of the cases above the invoked *interface* method extends the **`Shareable`** *interface*.

FMT_MSA.1 Management of security attributes

5.1.5.4 FMT_MSA.1.1/JCRE

The TSF shall enforce the ***FIREWALL access control SFP*** and the ***JCVM information flow control SFP*** to restrict the ability to ***modify the active context, the SELECTed applet Context and the ActiveApplets security attributes to the JCRE (S.JCRE).***

Note: The modification of the active context, SELECTed applet Context and ActiveApplets security attributes is performed in accordance with the rules given in [JCRE22], §4 ***and ([JCVM22], §3.4.***.

Additional Requirements on Logical Channels

FDP_RIP.1 Subset residual information protection

The element FDP_RIP.1.1/TRANSIENT must be substituted by the following one:

5.1.5.5 FDP_RIP.1.1/TRANSIENT

The TSF shall ensure that any previous information content of a resource is made unavailable upon the ***de-allocation of the resource*** from the following objects***: any transient object.***

Note: The events that provoke the de-allocation of any transient object are described in [JCRE22], *§5.1.*

Note: The clearing of **`CLEAR_ON_DESELECT`** objects is not necessarily performed when the owner of the objects is deselected. In the presence of multiselectable *applet* instances, **`CLEAR_ON_DESELECT`** memory segments may be attached to *applet*s that are active in different logical channels. Multiselectable *applet* instances within a same *package* must share the transient memory segment if they are concurrently active ([JCRE22], *§4.2*.

5.1.6 ODELG Security Functional Requirements

The following requirements are concerned with the secure deletion of information provoked by the object deletion mechanism. This mechanism is triggered by the applet who owns the deleted objects by invoking a specific API method.

FDP_RIP.1 Subset residual information protection

5.1.6.1 FDP_RIP.1.1/ODEL

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

Note: Freed data resources resulting from the invocation of the method **`javacard.framework.JCSystem.requestObjectDeletion()`** are reused. Requirements on *de-allocation* after the invocation of the method are described in [JCAPI22].

Note: There is no conflict with FDP_ROL.1.1 here because of the bounds on the rollback mechanism: the execution of **`requestObjectDeletion()`** is not in the scope of the rollback because it must be performed in between *APDU* command processing, and therefore no transaction can be in progress.

FPT_FLS.1 Failure with preservation of secure state

5.1.6.2 FPT_FLS.1.1/ODEL

The TSF shall preserve a secure state when the following type of failure occurs: ***the object deletion functions fail to delete all the unreferenced objects owned by the applet that requested the execution of the method***

Note: The TOE provides additional feedback information to the card manager in case of potential security violation (see *FAU_ARP.1*).

5.1.7 CarG Security Functional Requirements

This group of requirements applies to those configurations where the bytecode verifier is not embedded on the card. If this is the case, the TOE shall include requirements for preventing the installation of a *package* that has not been bytecode verified, or that has been modified after bytecode verification.

FCO_NRO.2 Enforced proof of origin

5.1.7.1 FCO_NRO.2.1/CM

The TSF shall enforce the generation of evidence of origin for transmitted ***application packages*** at all times.

Note: If this is the case and a new application package is received by the card for installation, the card manager first checks that it actually comes from the verification authority. The verification authority is the entity responsible for bytecode verification.

5.1.7.2 FCO_NRO.2.2/CM

The TSF shall be able to relate the identity of the originator of the information, and the ***application package*** contained in the information to which the evidence applies.

5.1.7.3 FCO_NRO.2.3/CM

The TSF shall provide a capability to verify the evidence of origin of information to ***the recipient*** given **at the time when a package is received.**

FIA_UID.1 Timing of identification

5.1.7.4 FIA_UID.1.1/CM

The TSF shall allow **the sending of the APDU commands to initiate communication through the trusted channel** on behalf of the user to be performed before the user is identified.

5.1.7.5 FIA_UID.1.2/CM

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

Note: package installation requires the user to be identified. Here by user is meant the one(s) defined in the component FMT_SMR.1/CM.

FDP_IFC.2 Complete information flow control

5.1.7.6 FDP_IFC.2.1/CM

The TSF shall enforce the ***PACKAGE LOADING information flow control SFP*** on ***S.CRD, S.BCV, S.SPY*** and all operations that cause that information to flow to and from subjects covered by the SFP.

Subjects (prefixed with an "*S*") covered by this policy are those involved in the reception of an application package by the card through a potentially unsafe communication channel:

Subject	Description
S.BCV	The subject representing who is in charge of the bytecode verification of the packages (also known as the verification authority).
S.CRD	The on-card entity in charge of package downloading.
S.SPY	Any other subject that may potentially intercept, modify, or permute the messages exchanged between the former two subjects.

The operations (prefixed with "*OP*") that make information to flow between the subjects are those enabling to send a message through and to receive a message from the communication channel linking the card to the outside world. It is assumed that any message sent through the channel as clear text can be read by the attacker. Moreover, the attacker may capture any message sent through the communication channel and send its own messages to the other subjects.

Operation	Description
OP.SEND(M)	A subject sends a message M through the communication channel.
OP.RECEIVE(M)	A subject receives a message M from the communication channel.

The information (prefixed with an "**I**") controlled by the typing policy is the *APDUs* exchanged by the subjects through the communication channel linking the card and the CAD. Each of those messages contain part of an application package that is required to be loaded on the card, as well as any control information used by the subjects (either ***S.BCV*** or ***S.SPY)*** in the communication protocol.

Information	Description
I.APDU	Any *APDU* sent to or from the card through the communication channel.

5.1.7.7 FDP_IFC.2.2/CM

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

FDP_IFF.1 Simple security attributes

5.1.7.8 FDP_IFF.1.1/CM

The TSF shall enforce the ***PACKAGE LOADING information flow control SFP*** based on the following types of subject and information security attributes:

(1) The keys used by the subjects S.BCV, S.CRD and S.CARDMANAGER acting on behalf of the card issuer to encrypt/decrypt and sign their messages;

(2) Authentication retry counter.

Note: The keys are implemented according to [VISA].

5.1.7.9 FDP_IFF.1.2/CM

The TSF shall permit an information flow between a controlled subject and controlled information through a controlled operation if the following rules hold:

(1) The subject S.CRD shall accept a message only if it comes from the subject S.CAD;

(2) The subject S.CRD shall accept an application package only if it has received all the APDUs sent by the subject S.CAD without modification and in the right order.

Note: Cap Files are checked for consistent format in addition to the check by the byte code verifier S.BCV.

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

5.1.7.10 FDP_IFF.1.3/CM

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

5.1.7.11 FDP_IFF.1.4/CM

The TSF shall provide **list of additional SFP capabilities: none.**

5.1.7.12 FDP_IFF.1.5/CM

The TSF shall explicitly authorize an information flow based on the following rules:

The information flow is authorised according the relevant rules in [VISA].

5.1.7.13 FDP_IFF.1.6/CM

The TSF shall explicitly deny an information flow based on the following rules:

The information flow is denied if the authentication retry counter limit is exceeded.

FDP_UIT.1 Data exchange integrity

These requirements apply to those configurations where bytecode verification is not considered as being part of the TOE. If this is the case, then the bytecode verifier can be seen as an external IT product, and *package*s to be loaded on the card are user data in transit from that external product to the *Java Card System*.

5.1.7.14 FDP_UIT.1.1/CM

The TSF shall enforce the ***PACKAGE LOADING information flow control SFP*** to be able to ***receive*** user data in a manner protected from ***modification, deletion, insertion and replay errors.***

> Note: Modification errors should be understood as modification, substitution, unrecoverable ordering change of data and any other integrity error that may cause the application package to be installed on the card to be different from the one sent by the CAD.

5.1.7.15 FDP_UIT.1.2/CM

The TSF shall be able to determine on receipt of user data, whether ***modification, deletion, insertion, replay of some of the pieces of the application sent by the CAD has occurred.***

FMT_MSA.1 Management of security attributes

5.1.7.16 FMT_MSA.1.1/CM

The TSF shall enforce the ***PACKAGE LOADING information flow control SFP*** to restrict the ability to **modify, delete, reset the security attributes: the keys used by the subjects to encrypt/decrypt and sign their messages and the authentication retry counter to the S.CARDMANAGER acting on behalf of the card issuer.**

FMT_MSA.3 Static attribute initialization

5.1.7.17 FMT_MSA.3.1/CM

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

5.1.7.18 FMT_MSA.3.2/CM

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

FMT_SMR.1 Security roles

5.1.7.19 FMT_SMR.1.1/CM

The TSF shall maintain the roles: ***S.CRD, S.BCV, S.SPY, S.CAD.***

5.1.7.20 FMT_SMR.1.2/CM

The TSF shall be able to associate users with roles.

FTP_ITC.1 Inter-TSF trusted channel

The following requirements apply to those configurations where bytecode verification is not considered as being part of the TOE. If this is the case, then the *CAD* can be seen as a remote IT product, and *package*s to be loaded on the card shall be transmitted using an inter-TSF trusted channel to prevent them from being modified during downloading. Such trusted channel connects the embedded *Java Card System* to the secured environment of the card issuer where the package has been verified.

5.1.7.21 FTP_ITC.1.1/CM

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

5.1.7.22 FTP_ITC.1.2/CM

The TSF shall permit ***the CAD placed in the card issuer secured environment*** to initiate communication through the trusted channel.

5.1.7.23 FTP_ITC.1.3/CM

The TSF shall initiate communication through the trusted channel for ***installing a new application package on the card.***

Note: there is no dynamic package loading on the Java Card platform. New packages can be installed on the card only on demand of the card issuer.

5.1.8 SCPG Security Functional Requirements

The following SFRs are related to the security requirements for the smart card platform, that is, operating system and chip that the Java Card System is implemented upon. The requirements are expressed in terms of security functional requirements from [CC2].

FCS_COP.1 Cryptographic operation

5.1.8.1 FCS_COP.1.1/SCP

The TSF shall **perform encryption and decryption** in accordance with a specified cryptographic algorithm **DES, RSA, AES** and cryptographic key sizes **of 64 bit, up to 4 kbit, (128, 192, 256) bit** that meet the following **list of standards: [DES], [RSA], [AES].**

Note: The FIPS PUB 46-3 and FIPS PUB 197 standards apply for the SCP dependent part of the DES and AES encryption and decryption, the Java Card API dependent part of the DES and AES encryption is compliant to [JCAPI22] (see: 5.1.1.2.5).

FPT_AMT.1 Abstract machine testing

5.1.8.2 FPT_AMT.1.1/SCP

The TSF shall run a suite of tests ***during initial start-up (at each power on)*** to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF.

> Note: The abstract machine that underlies the TSF comprises the lower levels of the SCP, that is, the OS and its dedicated native applications and/or APIs (for instance, hardware cryptographic functions/buffers), as well as the IC. Self-test of these components is, as an example, included in [PP0010]. These tests are initiated by the TSF of the SCP itself.

FPT_FLS.1 Failure with preservation of secure state

5.1.8.3 FPT_FLS.1.1/SCP

The TSF shall preserve a secure state when the following types of failures occur: **loss of power and card tearing.**

FRU_FLT.1 Degraded fault tolerance

5.1.8.4 FRU_FLT.1.1/SCP

The TSF shall ensure the operations of **memory management and cryptographic algorithms** when the following failures occur: **lack of RAM, lack of EEPROM, random generator failure.**

FPT_PHP.3 Resistance to physical attack

5.1.8.5 FPT_PHP.3.1/SCP

The TSF shall resist

(1) access to and modification *of the TOE hardware* using microelectronics tools, as:

> **Reverse Engineering,**
>
> **Disconnection and modification of security features,**
>
> **Access to sensitive information,**
>
> **Internal signal forcing**

(2) perturbation *of the TOE operations* by applying external sources of energy and resist operation the TOE outside the intended range of temperature, power and frequency, as:

> **a. Injection of faults in cryptographic computations, changing results of check operations (life cycles, bypass of authentication and PIN verifications),**
>
> **b. Alteration of processing and program flow in cryptographic computations resulting in unexpected behaviour such as instruction skip and address jump,**

c. Change of data internally used between software and hardware components of the TOE in cryptographic computations, obtaining of sensitive information in combination with other attacks, outside of the valid limits to the SCP

by responding automatically such that the TSP is not violated.

FPT_SEP.1 TSF domain separation

5.1.8.6 FPT_SEP.1.1/SCP

The TSF shall maintain a ***security domain*** for its own execution that protects it from interference and tampering by untrusted subjects.

5.1.8.7 FPT_SEP.1.2/SCP

The TSF shall enforce separation between the ***security domains*** of subjects in the TSC.

Note: The use of "security domain" here refers to execution space, and should not be confused with other meanings of security domains.

FPT_RVM.1 Non-bypass ability of the TSP

5.1.8.8 FPT_RVM.1.1/SCP

The TSF shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

Note: This component supports O.SCP.SUPPORT, which in turn contributes to the secure operation of the TOE, by ensuring that these latter and supporting platform security mechanisms cannot be bypassed.

Note: The TSF and TSC stated in these three components refer to that of the *SCP*.

FPT_RCV.3 Automated recovery without undue loss

5.1.8.9 FPT_RCV.3.1/SCP

Changed by [CCFI_056] to:

When automated recovery from **loss of power and card tearing** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

5.1.8.10 FPT_RCV.3.2/SCP

For **loss of power and card tearing** the TSF shall ensure the return of the TOE to a secure state using automated procedures.

5.1.8.11 FPT_RCV.3.3/SCP

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

5.1.8.12 FPT_RCV.3.4/SCP

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

FPT_RCV.4 Function recovery

5.1.8.13 FPT_RCV.4.1/SCP

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

5.1.9 CMGR Card Manager

The following SFRs are related to the security requirements for the card manager. These are requirements for the IT environment in [JCSPP], however for this ST they are requirements for the TOE. They are all expressed in terms of security functional requirements from [CC2].

FDP_ACC.1 Subset access control

5.1.9.1 FDP_ACC.1.1/CMGR

The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** on **S.CAD, S.CARDMANAGER, S.CRD, O.PACKAGE and O.APPLET and all operations among subjects and objects covered by the SFP.**

Subject/Object	Description
S.CAD	The CAD is involved and the starting point in any card content management operations (package loading, applet installation and applet and package deletion).
S.CARDMANAGER	Card Manager charges Installer and Applet Deletion Manager to perform card content management operations (content loading, installation and deletion).
S.CRD	The installer, which acts on behalf of the card issuer. This subject is involved in the loading of packages and installation of applets. It could play the role of the on-card entity in charge of package loading, which is involved in the PACKAGE LOADING security policy.
O.PACKAGE	Any Java Card package as code unit of card content loading and deletion.
O.APPLET	Any Java Card applet as unit of installation or deletion.

Table 6 Subjects/Objects for CARD CONTENT MANAGEMENT access control SFP

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

Operation	Description
OP.PACKAGE_LOADING(O.PACKAGE, package AID, load parameters, ...)	Load and link a package from the CAD into card non-volatile memory. Resource consumption is limited by the values in the load parameters
OP.APPLET_INSTALLATION(O.PACKAGE, O.APPLET, application AID, application privileges, ...)	Install and create an applet instance from an installed package with a specific application AID, application privileges and install parameters.
OP.APPLET_DELETION(applet instance AID)	Delete an applet instance.
OP.PACKAGE_DELETION(package AID)	Delete an installed package.

Table 7 Operations for CARD CONTENT MANAGEMENT access control SFP

Note: All these operations and their parameters are implemented according to [VISA].

FDP_ACF.1 Security attribute based access control

5.1.9.2 FDP_ACF.1.1/CMGR

Changed by [CCFI_103] to:

The TSF shall enforce the ***CARD CONTENT MANAGEMENT access control SFP*** to objects based on **the following: *S.CARDMANGER and* its *Card Life Cycle State and Security Level, S.CAD and S.CRD.***

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

Subject	Attributes
S.CARDMANAGER	Card Life Cycle State, Security Level
S.CAD	None
S.CRD	None

Table 8 Subject attributes

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

Name	Description
Card Life Cycle State	OP_READY, INITIALIZED, SECURED, CARD_LOCKED, TERMINATED
Security Level	NO_SECURITY_LEVEL, AUTHENTICATED, C_MAC, C_DECRYPTION

Table 9 Security attribute values

Card life cycle states and Secure Channel security levels are defined in [VISA]. The life cycle state of the card manager is the life cycle state of the card. The card life cycle states OP_READY and INITIALIZED are bound to the pre-issuance phase of the card, where the card is supposed to reside in the secure environment of the card issuer. The life cycle states SECURED and CARD_LOCKED are bound to the post-issuance phase of a card. The life cycle state TERMINATED may occur in either the pre-issuance phase or the post-issuance phase and is meant to indicate the end of the life cycle of the card.

No card content operations (loading, installing, deleting) are allowed in the life cycle states CARD_LOCKED or TERMINATED.

Prior to send card content management APDUs (LOAD; INSTALL, DELETE) from the terminal to the Card Manager, the terminal has to establish a Secure Channel to the Card Manager. The minimum security level for card content mamgement in card state OP_READY and INITIALIZED is AUTHENTICATED which mutually authenticates both entities, in card state SECURED is AUTHENTICATED plus C_MAC which mutually authenticates both entities and ensures message intergrity (see [VISA]).

The security level C_DECRYPTION that includes C_MAC can optionally be used to ensure confidentiality.

5.1.9.3 FDP_ACF.1.2/CMGR

The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **CARD CONTENT MANAGEMENT access control SFP:**

R.CMRG.1
Card content operations OP.PACKAGE_LOADING, OP.APPLET_INSTALLATION, OP.APPLET_DELETION and OP.PACKAGE_DELETION must only be performed by S.CAD and S.CARDMANAGER in card life cycle states OP_READY, INITIALIZED or SECURED.

R.CMRG.2

Card content operations OP.PACKAGE_LOADING, OP.APPLET_INSTALLATION, OP.APPLET_DELETION and OP.PACKAGE_DELETION must not be performed in card life cycle states CARD_LOCKED or TERMINATED.

R.CMRG.3
Card content operations OP.PACKAGE_LOADING, OP.APPLET_INSTALLATION, OP.APPLET_DELETION and OP.PACKAGE_DELETION in card life cycle states OP_READY or INITIALIZED must only be performed by S.CAD and S.CARDMANAGER after initiation of a Secure Channel with minimum security level AUTHENTICATED.

R.CMRG.4
Card content operations OP.PACKAGE_LOADING, OP.APPLET_INSTALLATION, OP.APPLET_DELETION and OP.PACKAGE_DELETION in card life cycle state SECURED must only be performed by S.CAD and S.CARDMANAGER after initiation of a Secure Channel with minimum security level C_MAC.

5.1.9.4 FDP_ACF.1.3/CMGR

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

5.1.9.5 FDP_ACF.1.4/CMGR

The TSF shall explicitly deny access of subjects to objects based on the rule: If the card life cycle state is TERMINATED or LOCKED the access of subjects for CARD CONTENT MANAGEMENT access control SFP on its objects is not allowed.

FMT_MSA.1 Management of security attributes

5.1.9.6 FMT_MSA.1.1/CMGR

The TSF shall enforce the ***CARD CONTENT MANAGEMENT access control SFP*** to restrict the ability to **modify** the security attributes: **Card Life Cycle State, Security Level** to **S.CARDMANAGER.**

FMT_SMF.1 Security management functions

5.1.9.7 FMT_SMF.1.1/CMGR

Changed by [CCFI_065]

The TSF shall be capable of performing the following security management functions:

Modification of the security attributes Card Life Cycle State and Security Level.

FMT_MSA.3 Static attribute initialisation

5.1.9.8 FMT_MSA.3.1/CMGR

The TSF shall enforce the ***CARD CONTENT MANAGEMENT access control SFP*** to provide ***restrictive*** default values for security attributes that are used to enforce the SFP.

5.1.9.9 FMT_MSA.3.2/CMGR

The TSF shall allow the **authorised identified roles: none** to specify alternative initial values to override the default values when an object or information is created.

FMT_SMR.1 Security roles

5.1.9.10 FMT_SMR.1.1/CMGR

The TSF shall maintain the roles: **S.CAD, S.CRD and S.CARDMANAGER.**

5.1.9.11 FMT_SMR.1.2/CMGR

The TSF shall be able to associate users with roles.

FIA_UID.1/CMGR Timing of identification

5.1.9.12 FIA_UID.1.1/CMGR

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

5.1.9.13 FIA_UID.1.2/CMGR

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

> Note: The list of TSF-mediated actions requires for the user attempting card content modifications to be identified. Here by user is meant the one(s) that in the Security Target shall be associated to the role(s) defined in the component FMT_SMR.1/CMGR

5.1.10 Further Functional Requirements not contained in [JCSPP]

The SFR in this section are not contained in [JCSPP].

FCS_RND.1 Quality metric for Random Numbers

5.1.10.1 FCS_RND.1.1

The TSF shall provide a mechanism to generate random numbers that meet the **class K3 of [AIS 20] with SOF-high**.

FPT_EMSEC.1 TOE Emanation

5.1.10.2 FPT_EMSEC.1.1

The TOE shall not emit **variations in power consumption or timing during command execution** in excess of **non-useful information** enabling access to **TSF data: D.JCS_KEYs and D.CRYPTO** and **User data: D.PIN, D.APP_KEYs**.

5.1.10.3 FPT_EMSEC.1.2

The TSF shall ensure **that unauthorized users** are unable to use the following interface **electrical contacts** to gain access to **TSF data: D.JCS_KEYs and D.CRYPTO** and **User data: D.PIN, D.APP_KEYs**.

FMT_LIM.1 Limited Capabilities

5.1.10.4 FMT_LIM.1.1

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

1. User Data to be disclosed or manipulated

2. TSF data to be disclosed or manipulated

3. software to be reconstructed and

4. substantial information about construction of TSF to be gathered which may enable other attacks.

FMT_LIM.2 Limited Availability

5.1.10.5 FMT_LIM.2.1

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

1. User Data to be disclosed or manipulated

2. TSF data to be disclosed or manipulated

3. software to be reconstructed and

4. substantial information about construction of TSF to be gathered which may enable other attacks.

5.2 TOE security assurance requirements

The security assurance requirement level is EAL4. The EAL is augmented with AVA_VLA.4 and ADV_IMP.2.

5.2.1 ACM Configuration management

5.2.1.1 ACM_AUT CM automation

ACM_AUT.1 Partial CM automation

5.2.1.1.1 **ACM_AUT.1.1D**

The developer shall use a CM system.

5.2.1.1.2 **ACM_AUT.1.2D**

The developer shall provide a CM plan.

5.2.1.1.3 **ACM_AUT.1.1C**

The CM system shall provide an automated means by which only authorised changes are made to the TOE implementation representation.

5.2.1.1.4 **ACM_AUT.1.2C**

The CM system shall provide an automated means to support the generation of the TOE.

5.2.1.1.5 **ACM_AUT.1.3C**

The CM plan shall describe the automated tools used in the CM system.

5.2.1.1.6 **ACM_AUT.1.4C**

The CM plan shall describe how the automated tools are used in the CM system.

5.2.1.1.7 **ACM_AUT.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.1.2 ACM_CAP CM capabilities

ACM_CAP.4 Generation support and acceptance procedures

5.2.1.2.1 **ACM_CAP.4.1D**

The developer shall provide a reference for the TOE.

5.2.1.2.2 **ACM_CAP.4.2D**

The developer shall use a CM system.

5.2.1.2.3 **ACM_CAP.4.3D**

The developer shall provide CM documentation.

5.2.1.2.4 ACM_CAP.4.1C

The reference for the TOE shall be unique to each version of the TOE.

5.2.1.2.5 ACM_CAP.4.2C

The TOE shall be labelled with its reference.

5.2.1.2.6 ACM_CAP.4.3C

The CM documentation shall include a configuration list, a CM plan, and an acceptance plan.

The configuration list shall uniquely identify all configuration items that comprise the TOE [CCFI_003].

5.2.1.2.7 ACM_CAP.4.4C

The configuration list shall describe the configuration items that comprise the TOE.

5.2.1.2.8 ACM_CAP.4.5C

The CM documentation shall describe the method used to uniquely identify the configuration items.

5.2.1.2.9 ACM_CAP.4.6C

The CM system shall uniquely identify all configuration items.

5.2.1.2.10 ACM_CAP.4.7C

The CM plan shall describe how the CM system is used.

5.2.1.2.11 ACM_CAP.4.8C

The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

5.2.1.2.12 ACM_CAP.4.9C

The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

5.2.1.2.13 ACM_CAP.4.10C

The CM system shall provide measures such that only authorised changes are made to the configuration items.

5.2.1.2.14 ACM_CAP.4.11C

The CM system shall support the generation of the TOE.

5.2.1.2.15 ACM_CAP.4.12C

The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

5.2.1.2.16 **ACM_CAP.4.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.1.3 ACM_SCP CM scope

ACM_SCP.2 Problem tracking CM coverage

5.2.1.3.1 **ACM_SCP.2.1D**

The developer shall provide a list of configuration items for the TOE [CCFI_004].

5.2.1.3.2 **ACM_SCP.2.1C**

The list of configuration items shall include the following: implementation representation; security flaws; and the evaluation evidence required by the assurance components in the ST [CCFI_004] .

5.2.1.3.3 **ACM_SCP.2.2C**

Deleted [CCFI_004].

5.2.1.3.4 **ACM_SCP.2.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2 ADO Delivery and operation

5.2.2.1 ADO_DEL Delivery

ADO_DEL.2 Detection of modification

5.2.2.1.1 **ADO_DEL.2.1D**

The developer shall document procedures for delivery of the TOE or parts of it to the user.

5.2.2.1.2 **ADO_DEL.2.2D**

The developer shall use the delivery procedures.

5.2.2.1.3 **ADO_DEL.2.1C**

The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

5.2.2.1.4 **ADO_DEL.2.2C**

The delivery documentation shall describe how the various procedures and technical measures provide for the detection of modifications, or any discrepancy between the developer's master copy and the version received at the user site.

5.2.2.1.5 **ADO_DEL.2.3C**

The delivery documentation shall describe how the various procedures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.

5.2.2.1.6 **ADO_DEL.2.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_IGS.1 Installation, generation, and start-up procedures

5.2.2.1.7 **ADO_IGS.1.1D**

The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

5.2.2.1.8 **ADO_IGS.1.1C**

The installation, generation and start-up documentation shall describe all the steps necessary for secure installation, generation, and start-up of the TOE [CCFI_051].

5.2.2.1.9 **ADO_IGS.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2.1.10 **ADO_IGS.1.2E**

The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.

5.2.3 ADV Development

5.2.3.1 ADV_FSP

ADV_FSP.2 Fully defined external interfaces

5.2.3.1.1 **ADV_FSP.2.1D**

The developer shall provide a functional specification.

5.2.3.1.2 **ADV_FSP.2.1C**

The functional specification shall describe the TSF and its external interfaces using an informal style.

5.2.3.1.3 **ADV_FSP.2.2C**

The functional specification shall be internally consistent.

5.2.3.1.4 **ADV_FSP.2.3C**

The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing complete details of all effects, exceptions and error messages.

5.2.3.1.5 **ADV_FSP.2.4C**

The functional specification shall completely represent the TSF.

5.2.3.1.6 **ADV_FSP.2.5C**

The functional specification shall include rationale that the TSF is completely represented.

5.2.3.1.7 **ADV_FSP.2.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.1.8 **ADV_FSP.2.2E**

The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.2 ADV_HLD High-level design

ADV_HLD.2 Security enforcing high-level design

5.2.3.2.1 **ADV_HLD.2.1D**

The developer shall provide the high-level design of the TSF.

5.2.3.2.2 **ADV_HLD.2.1C**

The presentation of the high-level design shall be informal.

5.2.3.2.3 **ADV_HLD.2.2C**

The high-level design shall be internally consistent.

5.2.3.2.4 **ADV_HLD.2.3C**

The high-level design shall describe the structure of the TSF in terms of subsystems.

5.2.3.2.5 **ADV_HLD.2.4C**

The high-level design shall describe the security functionality provided by each subsystem of the TSF.

5.2.3.2.6 **ADV_HLD.2.5C**

The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

5.2.3.2.7 **ADV_HLD.2.6C**

The high-level design shall identify all interfaces to the subsystems of the TSF.

5.2.3.2.8 **ADV_HLD.2.7C**

The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

5.2.3.2.9 **ADV_HLD.2.8C**

The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing details of effects, exceptions and error messages, as appropriate.

5.2.3.2.10 **ADV_HLD.2.9C**

The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

5.2.3.2.11 **ADV_HLD.2.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.2.12 **ADV_HLD.2.2E**

The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.3 ADV_IMP Implementation representation

See augmentation: 5.2.8.2 ADV_IMP.2 Implementation of the TSF

5.2.3.4 ADV_LLD

ADV_LLD.1 Low-level design

5.2.3.4.1 **ADV_LLD.1.1D**

The developer shall provide the low-level design of the TSF.

5.2.3.4.2 **ADV_LLD.1.1C**

The presentation of the low-level design shall be informal.

5.2.3.4.3 **ADV_LLD.1.2C**

The low-level design shall be internally consistent.

5.2.3.4.4 **ADV_LLD.1.3C**

The low-level design shall describe the TSF in terms of modules.

5.2.3.4.5 **ADV_LLD.1.4C**

The low-level design shall describe the purpose of each module.

5.2.3.4.6 **ADV_LLD.1.5C**

The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

5.2.3.4.7 **ADV_LLD.1.6C**

The low-level design shall describe how each TSP-enforcing function is provided.

5.2.3.4.8 **ADV_LLD.1.7C**

The low-level design shall identify all interfaces to the modules of the TSF.

5.2.3.4.9 **ADV_LLD.1.8C**

The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

5.2.3.4.10 **ADV_LLD.1.9C**

The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing details of effects, exceptions and error messages, as appropriate.

5.2.3.4.11 **ADV_LLD.1.10C**

The low-level design shall describe the separation of the TOE into TSP-enforcing and other modules.

5.2.3.4.12 **ADV_LLD.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.4.13 **ADV_LLD.1.2E**

The evaluator shall determine that the low-level design is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.5 ADV_RCR Representation correspondence

ADV_RCR.1 Informal correspondence demonstration

5.2.3.5.1 **ADV_RCR.1.1D**

The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

5.2.3.5.2 **ADV_RCR.1.1C**

For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

5.2.3.5.3 **ADV_RCR.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.6 ADV_SPM Security policy modelling

ADV_SPM.1 Informal TOE security policy model

5.2.3.6.1 **ADV_SPM.1.1D**

The developer shall provide a TSP model.

5.2.3.6.2 **ADV_SPM.1.2D**

The developer shall demonstrate correspondence between the functional specification and the TSP model.

5.2.3.6.3 **ADV_SPM.1.1C**

The TSP model shall be informal.

5.2.3.6.4 **ADV_SPM.1.2C**

The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modelled.

5.2.3.6.5 **ADV_SPM.1.3C**

The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modelled.

5.2.3.6.6 **ADV_SPM.1.4C**

The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

5.2.3.6.7 **ADV_SPM.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 AGD Guidance documents

5.2.4.1 AGD_ADM Administrator guidance

AGD_ADM.1 Administrator guidance

5.2.4.1.1 **AGD_ADM.1.1D**

The developer shall provide administrator guidance addressed to system administrative personnel.

5.2.4.1.2 **AGD_ADM.1.1C**

The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.

5.2.4.1.3 **AGD_ADM.1.2C**

The administrator guidance shall describe how to administer the TOE in a secure manner.

5.2.4.1.4 **AGD_ADM.1.3C**

The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

5.2.4.1.5 **AGD_ADM.1.4C**

The administrator guidance shall describe all assumptions regarding user behaviour that are relevant to secure operation of the TOE.

5.2.4.1.6 **AGD_ADM.1.5C**

The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.

5.2.4.1.7 **AGD_ADM.1.6C**

The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

5.2.4.1.8 **AGD_ADM.1.7C**

The administrator guidance shall be consistent with all other documentation supplied for evaluation.

5.2.4.1.9 **AGD_ADM.1.8C**

The administrator guidance shall describe all security requirements for the IT environment that are relevant to the administrator.

5.2.4.1.10 **AGD_ADM.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4.2 AGD_USR User guidance

AGD_USR.1 User guidance

5.2.4.2.1 **AGD_USR.1.1D**

The developer shall provide user guidance.

5.2.4.2.2 **AGD_USR.1.1C**

The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.

5.2.4.2.3 **AGD_USR.1.2C**

The user guidance shall describe the use of user-accessible security functions provided by the TOE.

5.2.4.2.4 **AGD_USR.1.3C**

The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.

5.2.4.2.5 **AGD_USR.1.4C**

The user guidance shall clearly present all user responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behaviour found in the statement of TOE security environment.

5.2.4.2.6 **AGD_USR.1.5C**

The user guidance shall be consistent with all other documentation supplied for evaluation.

5.2.4.2.7 **AGD_USR.1.6C**

The user guidance shall describe all security requirements for the IT environment that are relevant to the user.

5.2.4.2.8 **AGD_USR.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.5 ALC Life cycle support

5.2.5.1 ALC_DVS Development security

ALC_DVS.1 Identification of security measures

5.2.5.1.1 **ALC_DVS.1.1D**

The developer shall produce development security documentation.

5.2.5.1.2 **ALC_DVS.1.1C**

The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

5.2.5.1.3 **ALC_DVS.1.2C**

The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

5.2.5.1.4 **ALC_DVS.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.5.1.5 **ALC_DVS.1.2E**

The evaluator shall confirm that the security measures are being applied.

5.2.5.2 ALC_LCD Life cycle definition

ALC_LCD.1 Developer defined life-cycle model

5.2.5.2.1 **ALC_LCD.1.1D**

The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

5.2.5.2.2 **ALC_LCD.1.2D**

The developer shall provide life-cycle definition documentation.

5.2.5.2.3 **ALC_LCD.1.1C**

The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

5.2.5.2.4 **ALC_LCD.1.2C**

The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

5.2.5.2.5 **ALC_LCD.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.5.3 ALC_TAT Tools and techniques

ALC_TAT.1 Well-defined development tools

5.2.5.3.1 **ALC_TAT.1.1D**

The developer shall identify the development tools being used for the TOE.

5.2.5.3.2 **ALC_TAT.1.2D**

The developer shall document the selected implementation-dependent options of the development tools.

5.2.5.3.3 **ALC_TAT.1.1C**

All development tools used for implementation shall be well-defined.

5.2.5.3.4 **ALC_TAT.1.2C**

The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

5.2.5.3.5 **ALC_TAT.1.3C**

The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

5.2.5.3.6 **ALC_TAT.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6 ATE Tests

5.2.6.1 ATE_COV Coverage

ATE_COV.2 Analysis of coverage

5.2.6.1.1 **ATE_COV.2.1D**

The developer shall provide an analysis of the test coverage.

5.2.6.1.2 **ATE_COV.2.1C**

The analysis of the test coverage shall demonstrate the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.

5.2.6.1.3 **ATE_COV.2.2C**

The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.

5.2.6.1.4 **ATE_COV.2.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.2 ATE_DPT Depth

ATE_DPT.1 Testing: high-level design

5.2.6.3 ATE_DPT.1.1D

The developer shall provide the analysis of the depth of testing.

5.2.6.4 ATE_DPT.1.1C

The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design.

5.2.6.5 ATE_DPT.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.6 ATE_FUN Functional tests

ATE_FUN.1 Functional testing

5.2.6.6.1 **ATE_FUN.1.1D**

The developer shall test the TSF and document the results.

5.2.6.6.2 **ATE_FUN.1.2D**

The developer shall provide test documentation.

5.2.6.6.3 **ATE_FUN.1.1C**

The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

5.2.6.6.4 **ATE_FUN.1.2C**

The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

5.2.6.6.5 **ATE_FUN.1.3C**

The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.

5.2.6.6.6 **ATE_FUN.1.4C**

The expected test results shall show the anticipated outputs from a successful execution of the tests.

5.2.6.6.7 **ATE_FUN.1.5C**

The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

5.2.6.6.8 **ATE_FUN.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.7 ATE_IND Independent testing

ATE_IND.2 Independent testing - sample

5.2.6.7.1 **ATE_IND.2.1D**

The developer shall provide the TOE for testing.

5.2.6.7.2 **ATE_IND.2.1C**

The TOE shall be suitable for testing.

5.2.6.7.3 **ATE_IND.2.2C**

The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

5.2.6.7.4 **ATE_IND.2.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.7.5 **ATE_IND.2.2E**

The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

5.2.6.7.6 **ATE_IND.2.3E**

The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

5.2.7 AVA Vulnerability assessment

5.2.7.1 AVA_MSU Misuse

AVA_MSU.2 Validation of analysis

5.2.7.1.1 **AVA_MSU.2.1D**

The developer shall provide guidance documentation.

5.2.7.1.2 **AVA_MSU.2.2D**

The developer shall document an analysis of the guidance documentation.

5.2.7.1.3 **AVA_MSU.2.1C**

The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

5.2.7.1.4 **AVA_MSU.2.2C**

The guidance documentation shall be complete, clear, consistent and reasonable.

5.2.7.1.5 **AVA_MSU.2.3C**

The guidance documentation shall list all assumptions about the intended environment.

5.2.7.1.6 **AVA_MSU.2.4C**

The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

5.2.7.1.7 **AVA_MSU.2.5C**

The analysis documentation shall demonstrate that the guidance documentation is complete.

5.2.7.1.8 **AVA_MSU.2.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.7.1.9 **AVA_MSU.2.2E**

The evaluator shall repeat all configuration and installation procedures and other procedures selectively, to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

5.2.7.1.10 **AVA_MSU.2.3E**

The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

5.2.7.1.11 **AVA_MSU.2.4E**

The evaluator shall confirm that the analysis documentation shows that guidance is provided for secure operation in all modes of operation of the TOE.

5.2.7.2 AVA_VLA Vulnerability analysis

See augmentation: 5.2.8.1 AVA_VLA.4 Highly resistant

5.2.7.3 AVA_SOF Strength of TOE security functions

AVA_SOF.1 Strength of TOE security function evaluation

5.2.7.3.1 **AVA_SOF.1.1D**

The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.

5.2.7.3.2 **AVA_SOF.1.1C**

For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the PP/ST.

5.2.7.3.3 **AVA_SOF.1.2C**

For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST.

5.2.7.3.4 **AVA_SOF.1.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.7.3.5 **AVA_SOF.1.2E**

The evaluator shall confirm that the strength claims are correct.

5.2.8 Augmentations

5.2.8.1 AVA_VLA.4 Highly resistant

5.2.8.1.1 **AVA_VLA.4.1D**

The developer shall perform a vulnerability analysis [CCFI_051].

5.2.8.1.2 **AVA_VLA.4.2D**

The developer shall provide vulnerability analysis documentation [CCFI_051].

5.2.8.1.3 **AVA_VLA.4.1C**

The vulnerability analysis documentation shall describe the analysis of the TOE deliverables performed to search for ways in which a user can violate the TSP [CCFI_051].

5.2.8.1.4 **AVA_VLA.4.2C**

The vulnerability analysis documentation shall describe the disposition of identified vulnerabilities [CCFI_051].

5.2.8.1.5 **AVA_VLA.4.3C**

The vulnerability analysis documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE [CCFI_051].

5.2.8.1.6 **AVA_VLA.4.4C**

The vulnerability analysis documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks [CCFI_051].

5.2.8.1.7 **AVA_VLA.4.5C**

The vulnerability analysis documentation shall show that the search for vulnerabilities is systematic [CCFI_051].

5.2.8.1.8 AVA_VLA.4.6C

The vulnerability analysis documentation shall provide a justification that the analysis completely addresses the TOE deliverables [CCFI_051].

5.2.8.1.9 **AVA_VLA.4.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.8.1.10 **AVA_VLA.4.2E**

The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the identified vulnerabilities have been addressed.

5.2.8.1.11 **AVA_VLA.4.3E**

The evaluator shall perform an independent vulnerability analysis.

5.2.8.1.12 **AVA_VLA.4.4E**

The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.

5.2.8.1.13 **AVA_VLA.4.5E**

The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a high attack potential.

5.2.8.2 ADV_IMP.2 Implementation of the TSF

5.2.8.2.1 **ADV_IMP.2.1D**

The developer shall provide the implementation representation for the entire TSF.

5.2.8.2.2 **ADV_IMP.2.1C**

The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

5.2.8.2.3 **ADV_IMP.2.2C**

The implementation representation shall be internally consistent.

5.2.8.2.4 **ADV_IMP.2.3C**

The implementation representation shall describe the relationships between all portions of the implementation.

5.2.8.2.5 **ADV_IMP.2.1E**

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.8.2.6 **ADV_IMP.2.2E**

The evaluator shall determine that the implementation representation is an accurate and complete instantiation of the TOE security functional requirements.

5.3 Security requirements for the IT environment

5.3.1 Bytecode Verification

The following SFRs are related to bytecode verification. A bytecode verifier can be understood as a process that acts as a filter on a *CAP* file verifying that the bytecodes of the methods defined in the file conform to certain well-formed requirements.

FDP_IFC.2 Complete information flow control

5.3.1.1 FDP_IFC.2.1/BCV

The TSF shall enforce the ***TYPING information flow control SFP*** on ***S.LOCVAR, S.STCKPOS, S.FLD, S.MTHD*** and all operations that cause that information to flow to and from subjects covered by the SFP.

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

Subject	Description
S.LOCVAR	Any local variable of the currently executed method.
S.STCKPOS	Any operand stack position of the currently executed method.
S.FLD	Any field declared in a package loaded on the card.
S.MTHD	Any method declared in a package loaded on the card.

Table 10 Subjects for TYPING information flow control SFP

The operations (prefixed with "*OP*") that make information flow between the subjects are all bytecodes. For instance, the *aload_0* bytecode causes information to flow from the local variable 0 to the top of the operand stack; the bytecode *putfield(x)* makes information flow from the top of the operand stack to the field x; and the *return_a* bytecode makes information flow out of the currently executed method.

Operation	Description
OP.BYTECODE(BYTCD)	Any bytecode for the Java Card platform ("Java Card bytecode").

Table 11 Operation for TYPING information flow control SFP

The information (prefixed with an "I") controlled by the typing policy are the bytes, shorts, integers, references and return addresses contained in the different storage units of the JCVM (local variables, operand stack, static fields, instance fields and array positions).

Information	Description

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

Information	Description
I.BYTE(BY)	Any piece of information that can be encoded in a byte.
I.SHORT(SH)	Any piece of information that can be encoded in a short value.
I.INT(W1,W2)	Any piece of information that can be encoded in an integer value, which in turn is encoded in two words w1 and w2.
I.REFERENCE(RF)	Any reference to a class instance or an array.
I.ADDRESS(ADRS)	Any return address of a subroutine.

Table 12 Information controlled by the TYPING policy

5.3.1.2 FDP_IFC.2.2/BCV

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

FDP_IFF.2 Hierarchical security attributes

See FMT_MSA.1 for more information about security attributes.

5.3.1.3 FDP_IFF.2.1/BCV

The TSF shall enforce the ***TYPING information flow control SFP*** based on the following types of subject and information security attributes:

(1) type attribute of the information,

(2) type attribute of the storage units of the JCVM,

(3) class attribute of the fields and methods,

(4) bounds attribute of the methods.

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

Subject/Information	Attributes
S.LOCVAR	*TYPE*
S.STCKPOS	*TYPE*
S.FLD	*TYPE, CLASS*
S.MTHD	*TYPE, CLASS, BOUNDS*
I.BYTE(BY)	*TYPE*
I.SHORT(SH)	*TYPE*
I.INT(W1,W2)	*TYPE*
I.REFERENCE(RF)	*TYPE*
I.ADDRESS(ADRS)	*TYPE*

Table 13 Security for subjects/information of TYPING policy

The following table describes the security attributes.

Attribute Name	Description

Attribute Name	Description
TYPE	Either the type attached to the information, or the type held or declared by the subject.
CLASS	The class where a field or method is declared.
BOUNDS	The start and end of the method code inside the method component of the CAP file where it is declared.

Table 14 Security attributes description for TYPING policy

The *TYPE* security attribute attached to local variables and operand stack positions is the type of information they currently hold. The *TYPE* attribute of the fields and the methods is the type declared for them by the programmer.

The *BOUNDS* attribute of a method is used to prevent control flow to jump outside the currently executed method.

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

Name	Description
TYPE	byte, short, int_1, int_2, any class name *C*, *T*[] with *T* any type in the Java Card platform (“Java Card type”), T_0 (T_1 x_1, …. T_n x_n) with T_0,.. T_n any Java Card type, RetAddrs(adrs), Top, Null, ⊥.
CLASS	The name of a class, represented as a reference into the class Component of one of the packages loaded on the card.
BOUNDS	Two integers marking a rank into the method component of a package loaded on the card.

Table 15 Values of security attributes for TYPING policy

Byte values have type ***byte*** and short values have type ***short***. The first and second halves of an integer value has respectively type ***int_1***, and ***int_2***. The type of a reference to an instance of the class *C* is ***C*** itself**.** A reference to an array of elements of type *T* has type ***T[]***. From the previous basic types it is possible to build the type T_0 (T_1 x_1, …. T_n x_n) of a method. A return address *adrs* of a subroutine has type ***RetAddrss(adrs)***. Finally, the former Java Card types are extended with three extra types **Top**, **Null** and ⊥**,** so that the domain of types forms a complete lattice. **Top** is the type of any piece of data, that is, the maximum of the lattice. **Null** is the type of the default value null of all the reference types (classes and arrays). ⊥ is the type of an element that belongs to all types (for instance the value 0, provided that null is represented as zero).

5.3.1.4 FDP_IFF.2.2/BCV

The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules based on the ordering relationships between security attributes hold:

The following rules constitute a synthetic formulation of the information flow control:

R. JAVA.6 If the bytecode pushes values from the operand stack, then there are a sufficient number of values on the stack and the values of the attribute TYPE of the top positions of the stack is appropriate with respect to the ones expected by the bytecode.

R.JAVA.7 If the bytecode pushes values onto the operand stack, then there is sufficient room on the operand stack for the new values. The values, with the appropriate attribute TYPE value are added to the top of the operand stack.

R.JAVA.8 If the bytecode modifies a local variable with a value with attribute TYPE T, it must be recorded that the local variable now contains a value of that type. In addition, the variable shall be among the local variables of the method.

R.JAVA.9 If the bytecode reads a local variable, it must be ensured that the specified local variable contains a value with the attribute TYPE specified by the bytecode.

R.JAVA.10 If the bytecode uses a field, it must be ensured that its value is of an appropriate type. This type is indicated by the CLASS attribute of the field.

R.JAVA.11 If the bytecode modifies a field, then it must be ensured that the value to be assigned is of an appropriate type. This type is indicated by the CLASS attribute of the field

R.JAVA.12 If the bytecode is a method invocation, it must be ensured that it is invoked with arguments of the appropriate type. These types are indicated by the TYPE and CLASS attributes of the method.

R.JAVA.13 If the bytecode is a branching instruction, then the bytecode target must be defined within the BOUNDS of the method in which the branching instruction is defined.

Note: The rules described above are strongly inspired in the rules described in section 4.9 of [JVM], ***Second Edition***. The complete set of typing rules can be derived from the "Must" clauses from Chapter 7 of [JCVM22] as instances of the rules defined above.

5.3.1.5 FDP_IFF.2.3/BCV

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

5.3.1.6 FDP_IFF.2.4/BCV

The TSF shall provide the following list of additional SFP capabilities: ***none.***

5.3.1.7 FDP_IFF.2.5/BCV

The TSF shall explicitly authorize an information flow based on the following rules: ***none***.

5.3.1.8 FDP_IFF.2.6/BCV

The TSF shall explicitly deny an information flow based on the following rules: ***none.***

5.3.1.9 FDP_IFF.2.7/BCV

The TSF shall enforce the following relationships for any two valid information flow control security attributes:

a) There exists an ordering function that, given two valid security attributes, determines if the security attributes are equal, if one security

attribute is greater than the other, or if the security attributes are incomparable; and

b) ***There exists a least upper bound in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is greater than or equal to the two valid security attributes; and***

c) ***There exists a greatest lower bound in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is not greater than the two valid security attributes.***

Note: The order relationship between Java Card types is described, for instance, in the description of the **checkcast** bytecode of[JCVM22]. That relation is with the following rules:

- **Top** is the maximum of all types;
- **Null** is the minimum of all classes and array types;
- ⊥ is the minimum of all types.

These three extra types are introduced in order to satisfy the two last items in requirement FDP_IFF.2.7.

FMT_MSA.1 Management of security attributes

(See FMT_SMR.1.1/BCV for the roles)

5.3.1.10 FMT_MSA.1.1/BCV.1

The TSF shall enforce the ***TYPING information flow control SFP*** to restrict the ability to ***modify*** the ***TYPE security attribute of the fields and methods*** to ***none.***

5.3.1.11 FMT_MSA.1.1/BCV.2

The TSF shall enforce the ***TYPING information flow control SFP*** to restrict the ability to ***modify*** the ***TYPE security attribute of local variables and operand stack position*** to the role ***Bytecode Verifier.***

Note: The TYPE attribute of the local variables and the operand stack positions is identified to the attribute of the information they hold. Therefore, this security attribute is possibly modified as information flows. For instance, the rules of the typing function enable information to flow from a local variable *lv* to the operand stack by the operation `sload`, provided that the value of the type attribute of *lv* is ***short***. This operation hence modifies the type attribute of the top of the stack. The modification of the security attributes should is done according to the typing rules derived from ***Chapter 7 of*** [JCVM22].

Security management functions

5.3.1.12 FMT_SMF.1.1/BCV

Changed by [CCFI_065]

The TSF shall be capable of performing the following security management functions: ***modification of* the *TYPE security attribute of the fields and methods*** and ***modification of*** the ***TYPE security attribute of local variables and operand stack position*.**

FMT_MSA.2 Secure security attributes

5.3.1.13 FMT_MSA.2.1/BCV

The TSF shall ensure that only secure values are accepted for security attributes.

Note: During the type verification of a method, the bytecode verifier makes intensive use of the information provided in the CAP format like the sub-class relationship between the classes declared in the package, the type and class declared for each method and field, the rank of exceptions associated to each method, and so on. All that information can be thought of as security attributes used by the bytecode verifier, or as information relating security attributes. Moreover, the bytecode verifier relies on several properties about the CAP format. All the properties on the CAP format required by the bytecode verifier could, for instance, be completely described in the TSP model, and the bytecode verifier should ensure that they are satisfied before starting type verifications. Examples of such properties are:

- Correspondences between the different components of the CAP file (for instance, each class in the class component has an entry in the descriptor component).
- Pointer soundness (example: the index argument in a static method invocation always has an entry in the constant pool);
- Absence of hanged pointers (example: each exception handler points to the beginning of some bytecode);
- Redundant information (enabling different ways of searching for it);
- Conformance to the Java Language Specification respecting the access control features mentioned in §2.2 of [JCVM22].
- Packages that are loaded post-issuance can not contain native code.

FMT_MSA.3 Static attribute initialisation

5.3.1.14 FMT_MSA.3.1/BCV

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

Note: The TYPE attribute of the fields and methods is fixed by the application provider and never modified. When a method is invoked, the operand (type) stack is empty. The initial type assigned to those local variables that correspond to the method parameters is the type the application provider declared for those parameters. Any other local variable used in the method is set to the default value Top.

5.3.1.15 FMT_MSA.3.2/BCV

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

Note: The intent is to have none of the identified roles to have privileges with regards to the default values of the TYPE attributes.

FMT_SMR.1 Security roles

5.3.1.16 FMT_SMR.1.1/BCV

The TSF shall maintain the role: ***Bytecode Verifier.***

Note: the actual set of roles defined in the ST depends on the configuration.

5.3.1.17 FM**T_SMR.1.2/BCV**

The TSF shall be able to associate users with roles.

FRU_RSA.1 Maximum quotas

5.3.1.18 FRU_RSA.1.1/BCV

The TSF shall enforce maximum quotas of the following resources: ***the operand stack*** and ***the local variables*** that ***a method*** can use ***simultaneously.***

6 TOE summary specification

6.1 TOE security functions

The minimum strength for the security functions is SOF-high.

TOE Security Function	SOF claim	Description
SF.TRANSACTION	not appropriate	This TOE Security Function is not realised by a probabilistic or permutational noncryptographic mechanism.
SF.ACCESS_CONTROL	not appropriate	This TOE Security Function is not realised by a probabilistic or permutational noncryptographic mechanism.
SF.CRYPTO	high	The random number generator is a probabilistic mechanism. The deterministic random number generator is rated K3 (high) according to [AIS20].
SF.INTEGRITY	high	There is a probabilistic mechanism for the integrity check of checksummed data.
SF.SECURITY	not appropriate	This TOE Security Function is not realised by a probabilistic or permutational noncryptographic mechanism.
SF.APPLET	not appropriate	This TOE Security Function is not realised by a probabilistic or permutational noncryptographic mechanism.
SF.RMI	not appropriate	This TOE Security Function is not realised by a probabilistic or permutational noncryptographic mechanism.
SF.CARRIER	not appropriate	This TOE Security Function is not realised by a probabilistic or permutational noncryptographic mechanism.
SF.CARDMANAGER	not appropriate	This TOE Security Function is not realised by a probabilistic or permutational noncryptographic mechanism.

Table 16 SOF claims for TOE Security Functions

6.1.1 SF.TRANSACTION

This security function ensures the rollback process[13]. It provides assurance in the Java objects update in EEPROM.

1. The rollback operation restores the original values of the persistent objects (modified during the transaction) and clears the dedicated transaction area.

[13] Java Card technology supports a transaction mechanism with commit and rollback capability to guarantee that complex operations can be accomplished atomically; either they successfully complete or their partial results are not put into effect.

2. The TOE permits the rollback of the OP.JAVA, OP.CREATE on the O.JAVAOBJECTs.

6.1.2 SF.ACCESS_CONTROL

This security function is in charge of access control for the TOE. It is in charge of the FIREWALL access control SFP and the JCVM information flow control SFP.

1. The TOE maintains a security domain for its own execution that protects it from interference and tampering by untrusted subjects and enforces separation between the security domains of subjects in the TSC. The TOE enforces the firewall access control SFP. The TOE enforces the information flow SFP to control the flow of information between subjects (Table 2, Table 3).
2. The TOE restricts the ability to modify the list of registered applets and packages AID to the JCRE and maintains the following list of security attributes belonging to individual users: the AID and version number of each package, the AID of each registered applet, and whether a registered applet is currently selected for execution.
3. The TOE requires each user to identify itself before allowing any other TSF-mediated actions on behalf of that user and associates the following user security attributes with subjects acting on behalf of that user: Package AID, or “JCRE”.
4. Only secure values are accepted for security attributes.
5. When a Java object access contravenes the access rules defined in the 6.2 section of the document, this security function shall throw an exception.
6. The ability to modify the active context and the SELECTed applet Context security attributes is restricted to the JCRE.
7. The TOE provides Inter-TSF data consistency. The TOE uses rules stated in FPT_TDC.1.2 when interpreting the TSF data from another trusted IT product.

6.1.3 SF.CRYPTO

This security function controls all the operations related to the cryptographic key management and cryptographic operations.

This Security Function is composed of:

1. Key Generation for 3-DES/RSA-CRT/AES is done according to [JCAPI22].
2. Key access and distribution: the TOE provides 3-DES key (112,168 bit), RSA (1024 up to 2048 bit) and AES (128, 192, 256 bit) access (5.1.1.2.3) and distribution (5.1.1.2.2) in accordance with [JCAPI22].
3. Key destruction: the TOE provides a cryptographic 3-DES, RSA and AES key destruction method (5.1.1.2.4).

4. A cryptographic algorithm must be initialized with a key that corresponds to its type and which length is correct before use.
5. The TOE provides cryptographic operations (encryption and sign/verify) in accordance with [JCAPI22] that are relying in part on relevant libraries of the SCP (3-DES/RSA/AES).
6. Random number generation according to [AIS20] class K3 with SOF-high.

6.1.4 SF.INTEGRITY

This security function provides a means to check the integrity of checksummed data stored in EEPROM.

1. This security function initializes the checksum of cryptographic keys, PIN values and their associated security attributes.
2. The TOE monitors user data stored within the TSC for integrity errors by checksum testing.
3. Upon detection of a data integrity error on cryptographic keys, PIN values and their associated security attributes the TOE will throw an exception and prevent the usage of this key/PIN or switch to an endless loop. This is a secure state.
4. The TOE runs a suite of self-test during initial start-up.
5. The TOE provides authorized users with the capability to verify the integrity of stored TSF data and of stored TSF executable code.

6.1.5 SF.SECURITY

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

1. The TOE throws an exception, locks the card session or reinitialises the Java Card System and its JCRE data upon detection of a potential security violation and preserves a secure state. The SCP resists tampering of physical operating conditions (voltage supply, clock frequency and temperature) beyond the valid limits by responding automatically such that the SCP is in a secured state.
2. The TOE ensures the non-observability of operations on sensitive information like PINs and crytographic keys by relying on the firewall and other mechanism and additionally on methods of the SCP [for example countermeasures against SPA/DPA attacks] that will not leak information on PINs and crytographic keys in case of an attack.

3. The TOE ensures that any previous information content of a resource is made unavailable upon the allocation of the resource to class instances and arrays and the APDU buffer.
4. The TSF ensures that during command execution there are no usable variations in power consumption (measurable at e. g. electrical contacts) or timing (measurable at e. g. electrical contacts) that might disclose cryptographic keys or PINs. All functions of SF.CRYPTO are resistant to side-channel attacks (e.g. timing attack, SPA, DPA, DFA, EMA, DEMA).
5. The certified hardware (part of the TOE) provides Logical Protection (F.LOG) and protection of Mode Control (F.COMP) [STLCD40], [STLCD80], [STLCD144] and provides a hardware random number generation according to [AIS31].

6.1.6 SF.APPLET

This security function ensures the secure loading of a *package* or installing of an *applet by* S.CRD *and the secure* deletion of *applet*s and/or *packages by* S.ADEL.

1. The TOE enforces the FIREWALL access control SFP when loading of a *package* or installing of an *applet* and maintains the installer role.
2. The TOE uses the security attributes associated with the loaded *packages* or installed *applets*.
3. The *package*, loading is allowed by the TOE only if, for each dependent *package*, its *AID* attribute is equal to a resident package *AID* attribute, the major (minor) Version attribute associated to the former is equal (less than or equal) to the major (minor) Version attribute associated to the latter ([JCVM22],§4.5.2).
4. When the installer fails to load/install a package/applet it preserves a secure state as described in [JCRE22] §10.1.4. and enters a maintenance mode where the ability to return the TOE to a secure state is provided for reset, insufficient EEPROM, failure in cryptographic safeguarding, package references (versions) mismatching
5. For imported packages and declared classes, methods and fields that packages can use simultaneously the TOE shall enforce maximum quotas.
6. The TOE enforces the applet deletion access control SFP.
7. The TOE restricts the ability to modify the ActiveApplets security attribute to the JCRE.
8. The TOE ensures that any previous information content of a resource is made unavailable upon the de-allocation of the resource from applet instances and/or packages and from the objects owned by the context of an applet instance which triggered the execution of the method `javacard.framework.JCSystem.requestObjectDeletion()`.
9. The TOE preserves a secure state when the applet deletion manager fails to delete a package/applet as described in [JCRE22], §11.3.4 and the object deletion functions

fail to delete all the unreferenced objects owned by the applet that requested the execution of the method.

6.1.7 SF.RMI

This security function ensures secure remote method invocation features, which provides a new protocol of communication between the terminal and the applets.

1. The TOE enforces the JCRMI access control SFP to control the access to remote objects when the RMI service is used.
2. The TOE enforces the JCRMI information flow control SFP to control the flow of information that takes place when the RMI service is used.
3. The TOE enforces the FIREWALL access control SFP and the FIREWALL information flow control SFP to restrict the ability to modify
   - the ActiveApplets security attribute to the JCRE,
   - the security attribute Exported of an O.REMOTE_OBJ to its owner,
   - the security attribute Returned References of an O.RMI_SERVICE to its owner

   and provides restrictive default values for security attributes that are used to enforce the SFP.
4. The TOE restricts the ability to revoke the Returned References security attribute of an O.RMI_SERVICE to the JCRE.
5. The TOE enforces the rules that determine the lifetime of remote object references.
6. The TOE maintains the role applet and associates users with this role.

6.1.8 SF.CARRIER

This security function ensures secure downloading of applications on the card.

1. The TOE enforces the generation of evidence of origin for transmitted application packages at all times.
2. The TOE is able to relate the identity of the originator of the information, and the application package contained in the information to which the evidence applies.
3. The TOE provides a capability to verify the evidence of origin of information to the recipient given at the time it is received.
4. The TOE allows the sending of the APDU commands to initiate communication through the trusted channel on behalf of the user to be performed before the user is identified.

5. The TOE requires each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.
6. The TOE enforces the PACKAGE LOADING information flow control SFP to secure the reception of an application package by the card through a potentially unsafe communication channel.
7. The TOE enforces the PACKAGE LOADING information flow control SFP to provide restrictive default values for security attributes that are used to enforce the SFP.
8. The TOE maintains the roles: S.CRD, S.BCV, S.SPY, S.CAD and associates users with these roles.
9. The TOE provides a communication channel between itself and a remote IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.
10. The TOE permits the CAD placed in the card issuer secured environment to initiate communication through the trusted channel.
11. The TOE requires communication through the trusted channel for installing a new application package on the card.

6.1.9 SF.CARDMANAGER

This security function ensures the security for the card manager.

1. The TOE enforces the CARD CONTENT MANAGEMENT access control SFP on S.CAD, S.CARDMANAGER, S.CRD, O.PACKAGE and O.APPLET and all operations among subjects and objects covered by the SFP and to objects based on S.CARDMANGER and its Card Life Cycle State and Security Level, S.CAD and S.CRD.
2. The TOE enforces the CARD CONTENT MANAGEMENT access control SFP to restrict the ability to modify the security attributes: Card Life Cycle State, Security Level to S.CARDMANAGER.
3. The TOE is capable of the modification of the security attributes Card Life Cycle State and Security Level.
4. The TOE enforces the CARD CONTENT MANAGEMENT access control SFP to provide restrictive default values for security attributes that are used to enforce the SFP.
5. The TOE maintains the roles S.CAD, S.CRD and S.CARDMANAGER.

6. The TOE allows GET DATA, INITIALIZE UPDATE and EXTERNAL AUTHENTICATE according to [VISA] on behalf of the user to be performed before the user is identified.
7. The TOE requires S.CAD, S.CRD and S.CARDMANAGER to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

6.2 Assurance measures

6.2.1 AM_ACM

The configuration management is described in SMARTCAFE_ACM.

6.2.2 AM_ADO

The delivery, installation, generation and start-up of the TOE are described in SMARTCAFE_ADO.

6.2.3 AM_ADV

The representation of the TSF is described in SMARTCAFE_ADV_SPM for security policy modelling, in SMARTCAFE_ADV_FSP for functional specification, in SMARTCAFE_ADV_HLD for high level design, in SMARTCAFE_ADV_LLD for low level design, in SMARTCAFE_ADV_IMP for implementation representation and in SMARTCAFE_ADV_RCR for representation correspondence.

6.2.4 AM_AGD

The guidance documentation is described in SMARTCAFE_AGD_USR for the user and in SMARTCAFE_AGD_ADM for the administrator.

6.2.5 AM_ALC

The life cycle support of the TOE during its development and maintenance is described in SMARTCAFE_ALC.

6.2.6 AM_ATE

The testing of the TOE is described in SMARTCAFE_ATE.

6.2.7 AM_AVA

The vulnerability assessment for the TOE is described in SMARTCAFE_AVA_MSU for the misuse, in SMARTCAFE_AVA_SOF for the strength of TOE security functions and in SMARTCAFE_AVA_VLA for the vulnerability analysis.

7 PP claims

7.1 PP Reference

This security target (ST) is based on the following protection profile:

- Java Card System – Standard 2.2 Configuration Protection Profile, Version: 1.0b, August 2003 [JCSPP]

This ST makes claims for formal conformance to this PP, as the ST fulfils all requirements of [JCSPP]. This ST even chooses a hierarchically higher augmentation of EAL4, in comparison to [JCSPP], by selecting ADV_IMP.2 and AVA_VLA.4.

Further assumptions, threats, one organizational security policy, security objectives, and IT security requirements not contained in [JCSPP] were defined in this ST and marked that they were not taken from [JCSPP].

7.2 PP Additions and Refinements

The following SFRs have been added compared to SFR for the TOE defined in the *Java Card System – Standard 2.2 Configuration Protection Profile* [JCSPP] :

- from **[JCSPP] SCP group**: FPT_AMT.1/SCP, FPT_FLS.1/SCP, FRU_FLT.2/SCP, FPT_PHP.3/SCP, FPT_SEP.1/SCP, FPT_RCV.3/SCP, FPT_RCV.4/SCP and FPT_RVM.1/SCP
- from **[JCSPP] CMGR group**: FDP_ACC.1/CMGR, FDP_ACF.1/CMGR, FMT_MSA.1/CMGR, FMT_MSA.3/CMGR, FMT_SMR.1/CMGR, and FIA_UID.1/CMGR
- from **[JCSPP] Core group**: FCS_COP.1.1/SCP
- from **[EAC]**: FPT_RND.1, FPT_EMSEC.1, FMT_LIM.1, FMT_LIM.2.

The following SFRs have been changed according to Common Criteria final interpretations compared to the SFR for the TOE defined in the *Java Card System – Standard 2.2 Configuration Protection Profile* [JCSPP] :

- FDP_IFF.1.1/JCVM
  changed by [CCFI_104]
- FMT_SMF.1.1/JCRE
  changed by [CCFI_065]
- FIA_USB.1.1
  changed by [CCFI_137]

- FIA_USB.1.2
  changed by [CCFI_137]
- FIA_USB.1.3
  changed by [CCFI_137]
- FDP_ACF.1.1/ADEL
  changed by [CCFI_103]
- FPT_RCV.3.1/SCP
  changed by [CCFI_056]
- FDP_ACF.1.1/CMGR
  changed by [CCFI_103]
- FMT_SMF.1.1/CMGR
  changed by [CCFI_065]
- FMT_SMF.1.1/BCV
  changed by [CCFI_065]

8 Rationale

The tables in sub-sections 8.1.3 Rationale tables of security objectives and security requirements provide the mapping of the security objectives and security requirements for the TOE.

8.1 Security objectives rationale

8.1.1 Threats

8.1.1.1 T.CONFID-JCS-CODE

Threats concerning the integrity and confidentiality of code are countered by the list of properties described in the (#.VERIFICATION) security issue. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of visibility. As none of those instructions enables to read or modify a piece of code, no Java Card applet can therefore be executed to disclose or modify a piece of code. The (#.VERIFICATION) security issue is addressed in this configuration by the objective for the environment OE.VERIFICATION.

This threat is also addressed by the O.CARD_MANAGEMENT due to the implementation of the card issuer’s policy on the card by the card manager.

8.1.1.2 T.CONFID-JCS-DATA

Threats concerning the integrity and confidentiality of code are countered by the list of properties described in the (#.VERIFICATION) security issue. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of visibility. As none of those instructions enables to read or modify a piece of code, no Java Card applet can therefore be executed to disclose or modify a piece of code. The (#.VERIFICATION) security issue is addressed in this configuration by the objective for the environment OE.VERIFICATION.

The threats concerning confidentiality and integrity of data are countered by bytecode verification and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective. As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.

This threat is also addressed by the O.CARD_MANAGEMENT due to the card manager implementation of the card issuer’s policy on the card.

O.SCP.RECOVERY allows the TOE to eventually complete interrupted operations successfully, or recover to a consistent and secure state (#.SCP.1).

This threat is additionally addressed by the O.SCP.SUPPORT that controls the access to information proper of the TSFs.

8.1.1.3 T.INTEG-APPLI-CODE

Threats concerning the integrity and confidentiality of code are countered by the list of properties described in the (#.VERIFICATION) security issue. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of visibility. As none of those instructions enables to read or modify a piece of code, no Java Card applet can therefore be executed to disclose or modify a piece of code. The (#.VERIFICATION) security issue is addressed in this configuration by the objective for the environment OE.VERIFICATION.

This threat is also addressed by the O.CARD_MANAGEMENT due to the implementation of the card issuer’s policy on the card by the card manager.

8.1.1.4 T.INTEG-JCS-CODE

Threats concerning the integrity and confidentiality of code are countered by the list of properties described in the (#.VERIFICATION) security issue. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of visibility. As none of those instructions enables to read or modify a piece of code, no Java Card applet can therefore be executed to disclose or modify a piece of code. The (#.VERIFICATION) security issue is addressed in this configuration by the objective for the environment OE.VERIFICATION.

This threat is also addressed by the O.CARD_MANAGEMENT due to the implementation of the card issuer’s policy on the card by the card manager.

8.1.1.5 T.INTEG-APPLI-DATA

Threats concerning the integrity and confidentiality of code are countered by the list of properties described in the (#.VERIFICATION) security issue. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of visibility. As none of those instructions enables to read or modify a piece of code, no Java Card applet can therefore be executed to disclose or modify a piece of code. The (#.VERIFICATION) security issue is addressed in this configuration by the objective for the environment

OE.VERIFICATION. The threats concerning confidentiality and integrity of data are countered by bytecode verification and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective. As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken. Concerning the confidentiality and integrity of application sensitive data, as 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 that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, and O.FIREWALL).

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.
This threat is also addressed by the O.CARD_MANAGEMENT due to the implementation of the card issuer's policy on the card by the card manager.
O.SCP.RECOVERY allows the TOE to eventually complete interrupted operations successfully, or recover to a consistent and secure state (#.SCP.1).
This threat is additionally addressed by the O.SCP.SUPPORT that controls the access to information proper of the TSFs.
This threat is addressed by O.SHRD_VAR_INTEG that ensures that only the currently selected application may grant write access to a data memory area that is shared by all applications.

8.1.1.6 T.INTEG-JCS-DATA

Threats concerning the integrity and confidentiality of code are countered by the list of properties described in the (#.VERIFICATION) security issue. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of visibility. As none of those instructions enables to read or modify a piece of code, no Java Card applet can therefore be executed to disclose or modify a piece of code. The (#.VERIFICATION) security issue is addressed in this configuration by the objective for the environment OE.VERIFICATION. The threats concerning confidentiality and integrity of data are countered by bytecode verification and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective. As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken. If the PIN class of the Java Card API is used, the objective (O.FIREWALL) is also concerned.

This threat is also addressed by the O.CARD_MANAGEMENT due to the implementation of the card issuer's policy on the card by the card manager. O.SCP.RECOVERY allows the TOE to eventually complete interrupted operations successfully, or recover to a consistent and secure state (#.SCP.1).
This threat is additionally addressed by the O.SCP.SUPPORT that controls the access to information proper of the TSFs.

8.1.1.7 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. Note that the AIDs, which are used for applet identification, are TSF data.

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

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

8.1.1.8 T.SID.2

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

The objective O.INSTALL contributes to counter this threat for what relates to the critical phase of *applet* installation (because the *installer* may have special rights).

8.1.1.9 T.EXE-CODE.1

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

8.1.1.10 T.EXE-CODE.2

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

8.1.1.11 T.NATIVE

An applet tries to execute a native method to bypass some security function such as the firewall. A Java Card technology-based applet (“Java Card applet”) can only access native methods indirectly (O.NATIVE) that is, through an API which is assumed to be secure (A.NATIVE). 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).

An application cannot download its own native code on the card; see the objective *OE.APPLET*, which also contributes to enforce the objective countering this threat (O.NATIVE).

8.1.1.12 T.CONFID-APPLI-DATA

Threats concerning the integrity and confidentiality of code are countered by the list of properties described in the (#.VERIFICATION) security issue. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of visibility. As none of those instructions enables to read or modify a piece of code, no Java Card applet can therefore be executed to disclose or modify a piece of code. The (#.VERIFICATION) security issue is addressed in this configuration by the objective for the environment OE.VERIFICATION. The threats concerning confidentiality and integrity of data are countered by bytecode verification and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective. As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken. Concerning the confidentiality and integrity of application sensitive data, as 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 that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, and O.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) is also concerned.

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.
This threat is also addressed by the O.CARD_MANAGEMENT due to the implementation of the card issuer's policy on the card by the card manager.
O.SCP.RECOVERY allows the TOE to eventually complete interrupted operations successfully, or recover to a consistent and secure state (#.SCP.1).
This threat is additionally addressed by the O.SCP.SUPPORT that controls the access to information proper of the TSFs.
This threat is addressed by O.SHRD_VAR_CONFID that ensures that any data container that is shared by all applications is always cleaned after the execution of an application.

8.1.1.13 T.RESOURCES

An attacker prevents correct operation of the *Java Card System* through consumption of some resources of the card. This is directly countered by objectives on resource-management (O.RESOURCES) for runtime purposes and good working order (O.OPERATE) in a general manner.

In this configuration, consumption of resources during installation and other card management operations are covered, in case of failure, by O.INSTALL.

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

8.1.1.14 T.PHYSICAL

Covered by O.SCP.IC. Physical protections rely on the SCP and are therefore a TOE issue.

8.1.1.15 T.INSTALL

The attacker fraudulently **installs** an *applet* on the card post issuance. This threat is covered by the O.INSTALL and O.LOAD security objectives.

The objective O.CARD-MANAGEMENT supports OE.VERIFICATION and contributes to cover all the threats on confidentiality and integrity of code and data, the *T.INSTALL* threat, and the *T.INTEG-APPLI-CODE.2* and *T.INTEG-APPLI-DATA.2* threats. The objective also contributes, by preventing usurpation of identity resulting from a malicious installation of an applet on the card, to counter the threat T.SID.1.

8.1.1.16 T.INTEG-APPLI-CODE.2

The attacker modifies (part of) its own or another application code when an application *package* is transmitted to the card for installation. In this configuration the integrity of a package’s code is covered by the objective O.LOAD.

8.1.1.17 T.INTEG-APPLI-DATA.2

The attacker modifies (part of) the initialization data contained in an application *package* when the package is transmitted to the card for installation. In this configuration the integrity of a package’s code is covered by the objective O.LOAD.

8.1.1.18 T.EXE-CODE-REMOTE

The *O.REMOTE* security objective contributes to prevent the invocation of a method that is not supposed to be accessible from outside the card.

8.1.1.19 T.DELETION

This threat is covered by the O.DELETION security objective.

8.1.1.20 T.OBJ-DELETION

The objective O.CARD-MANAGEMENT supports OE.VERIFICATION and contributes to cover all the threats on confidentiality and integrity of code and data, the T.INSTALL threat, the T.DELETION threat and the T.INTEG-APPLI-CODE.2 and T.INTEG-APPLI-DATA.2 threats. The objective also contributes, by preventing usurpation of identity resulting from a malicious installation of an applet on the card, to counter the threat T.SID.1.

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

8.1.1.21 T.RND

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

8.1.1.22 T.LEAKAGE

O.SIDE_CHANNEL protects the security critical parts of the TOE from disclosure by interpretation of physical or logical behavior based on leakage observation (e. g. side channel attacks).

8.1.1.23 T.CHIP

O.CHECK_INIT prevents unauthorise installation of smartcard ICs that are not correctly tested and pre-personalised by the Chip Manufacturer by the check of the FabKey data that is part of the pre-personalisation data.

8.1.2 Assumptions

8.1.2.1 A.NATIVE

In this configuration all the security objectives directly or indirectly depend on the behaviour of the native code embedded on the card. This trusted native code is not subject to change during the lifetime of the card. The objective OE.NATIVE ensures that the environmental assumption A.NATIVE is upheld.

8.1.2.2 A.VERIFICATION

The objective OE.VERIFICATION upholds the assumption A.VERIFICATION.

8.1.2.3 A.APPLET

The objective *OE.APPLET* covers the assumption *A.APPLET*, and contributes to the enforcement of the objective O.NATIVE in the presence of post-issuance downloaded applications.

8.1.3 Rationale tables of environment elements and security objectives

	T.CONFID-JCS-DATA	T.INTEG-APPLI-CODE.2	T.INSTALL	T.INTEG-JCS-CODE	T.CHIP
O.SID	X				
O.OPERATE	X				
O.FIREWALL	X				
O.ALARM	X				
O.INSTALL			X		
O.LOAD		X	X		
O.CARD-MANAGEMENT			X		
O.CHECK_INIT					X

Table 17 Threats and security objectives rationale **(part 1/6)**

	T.PHYSICAL	T.CONFID-JCS-CODE	T.CONFID-JCS-DATA	T.INTEG-APPLI-CODE	T.INTEG-JCS-CODE
O.SCP.RECOVERY			X		
O.SCP.SUPPORT			X		
O.SCP.IC	X				
O.CARD-MANAGEMENT		X	X	X	X
OE.VERIFICATION		X	X	X	X

Table 18 Threats and security objectives rationale **(part 2/6)**

	T.INTEG-APPLI-DATA	T.INTEG-APPLI-DATA.2	T.INTEG-JCS-DATA	T.SID.1	T.SID.2	T.EXE-CODE.1
O.SID	X		X	X	X	
O.OPERATE	X		X		X	
O.RESOURCES						
O.FIREWALL	X		X	X	X	X
O.PIN-MNGT	X					
O.REALLOCATION	X					
O.SHRD_VAR_INTEG	X			X		
O.ALARM	X		X			
O.SHRD_VAR_CONFID				X		
O.LOAD		X				
O.TRANSACTION	X					
O.INSTALL				X	X	

Table 19 Threats and security objectives rationale **(part 3/6)**

	T.INTEG-APPLI-DATA	T.INTEG-JCS-DATA	T.SID.1	T.SID.2	T.EXE-CODE.1
O.CIPHER	X				

	T.INTEG-APPLI-DATA	T.INTEG-JCS-DATA	T.SID.1	T.SID.2	T.EXE-CODE.1
O.KEY-MNGT	X				
O.SCP.RECOVERY	X	X		X	
O.SCP.SUPPORT	X	X		X	
O.CARD-MANAGEMENT	X	X	X		
O.INSTALL				X	
OE.VERIFICATION	X	X			X

Table 20 Threats and security objectives rationale (part 4/6)

	T.NATIVE	T.CONFID-APPLI-DATA	T.DELETION	T.RESOURCES	T.RND	T.LEAKAGE
O.SID		X				
O.OPERATE		X		X		
O.RESOURCES				X		
O.FIREWALL		X				
O.PIN-MNGT		X				
O.NATIVE	X					
O.REALLOCATION		X				
O.SHRD_VAR_CONFID		X				
O.DELETION			X			
O.CARD-MANAGEMENT			X			
O.ALARM		X				
O.TRANSACTION		X				
O.RND					X	
O.SIDE_CHANNEL						X
O.INSTALL				X		

Table 21 Threats and security objectives rationale (part 5/6)

	T.EXE-CODE.2	T.NATIVE	T.CONFID-APPLI-DATA	T.RESOURCES	T.EXE-CODE-REMOTE	T.OBJ-DELETION
O.CIPHER			X			
O.KEY-MNGT			X			
O.SCP.RECOVERY			X	X		

O.SCP.SUPPORT			X	X		
O.CARD-MANAGEMENT			X			
OE.VERIFICATION	X	X	X			
OE.APPLET		X				
O.REMOTE					X	
O.OBJ-DELETION						X

Table 22 Threats and security objectives rationale (part 6/6)

	A.NATIVE	A.APPLET	A.VERIFICATION
OE.NATIVE	X		
OE.APPLET		X	
OE.VERIFICATION			X

Table 23 Assumptions and security objectives rationale

8.1.4 Organizational Policies Related to Security Objectives

Only one organizational security policy, OSP.VERIFICATION, has been defined for this configuration. This policy is covered by the security objective of the environment OE.VERIFICATION.

8.2 Security requirements rationale

8.2.1 Objectives

8.2.1.1 Security objectives for the TOE

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

- O.SCP.IC
- O.SCP.RECOVERY
- O.SCP.SUPPORT
- O.CARD-MANAGEMENT

8.2.1.1.1 O.SID

Subjects' identity is *AID*-based (*applet*s, *package*s), and is met by FDP_ITC.2, FIA_ATD.1, FMT_MSA.1, FMT_MSA.3, FMT_MTD.1 and FMT_MTD.3. Additional support includes FPT_RVM.1 and FPT_SEP.1.

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, FIA_USB.1).

O.INSTALL

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

8.2.1.1.2 O.LOAD

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

8.2.1.1.3 O.DELETION

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

8.2.1.1.4 O.OPERATE

The TOE is protected in various ways against applets actions (FPT_RVM.1, FPT_SEP.1, FPT_TDC.1), the FIREWALL access control policy (FDP_ACC.2, FDP_ACF.1 (FDP_ACF.1/FIREWALL, FDP_ACC.2/FIREWALL)), and is able to detect and block various failures or security violations during usual working (FPT_FLS.1, FAU_ARP.1 (FAU_ARP.1/JCS)). Startup of the TOE is covered by FPT_TST.1, and indirectly by FPT_AMT.1.

Its security-critical parts and procedures are also protected: safe recovery from failure is ensured (*FPT_RCV.3*), *applets'* installation may be cleanly aborted (FDP_ROL.1), communication with external users and their internal subjects is well-controlled (FDP_ITC.2, FIA_ATD.1, FIA_USB.1) 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.

8.2.1.1.5 **O.RESOURCES**

The TSFs detect stack/memory overflows during execution of applications (FAU_ARP.1, FRU_RSA.1, FPT_FLS.1). Failed installations are not to create memory leaks (FDP_ROL.1, FPT_RCV.3) as well. Memory management is controlled by the TSF (FMT_MTD.1, FMT_MTD.3, FMT_SMR.1) and is only accessible to user-applications through the API (FPT_RVM.1).

8.2.1.1.6 **O.FIREWALL**

This objective is met by the *Firewall access control policy* (*FDP_ACC.2*, *FDP_ACF.1*), the *JCVM information flow control policy* (FDP_IFF.1, FDP_IFC.1) the *JCRMI access control policy* (*FDP_ACC.2/JCRMI*, *FDP_ACF.1/JCRMI*) and the functional requirements FPT_RVM.1, FPT_SEP.1 and FDP_ITC.2. The functional requirements of the class *FMT* also indirectly contribute to meet this objective.

8.2.1.1.7 **O.NATIVE**

The *JCVM* is the machine running the bytecode of the *applet*s (FPT_RVM.1). These can only be linked with API methods or other *package*s already on the card. This objective mainly relies on the environmental objectives *OE.NATIVE* and *OE.APPLET*, which uphold the assumptions *A.NATIVE* and *A.APPLET* respectively.

8.2.1.1.8 **O.REALLOCATION**

The security objective is satisfied by FDP_RIP.1 (FDP_RIP.1/bArray, FDP_RIP.1/APDU, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, and FDP_RIP.1/OBJECTS) which imposes that the contents of the re-allocated block shall always be cleared before delivering the block.

8.2.1.1.9 **O.SHRD_VAR_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 (FDP_RIP.1.1/APDU, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/OBJECTS and FDP_RIP.1.1/bArray respectively). The JCVM information flow control policy (FDP_IFF.1 (FDP_IFF.1/JCVM), FDP_IFC.1 (FDP_IFC.1/JCVM)) prevents an application from keeping a pointer to a shared buffer, which could be used to read its contents when the buffer is being used by another application.

Protection of the array parameters of remotely invoked methods, which are global as well, is covered by the general initialization of method parameters (FDP_RIP.1).

8.2.1.1.10 **O.SHRD_VAR_INTEG**

This objective is met by the JCVM information flow control policy (FDP_IFF.1 (FDP_IFF.1/JCVM) and FDP_IFC.1 (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.

8.2.1.1.11 **O.ALARM**

This objective is met by FPT_FLS.1 (FPT_FLS.1/JCS) and FAU_ARP.1 (FAU_ARP.1/JCS) (see notes).

8.2.1.1.12 **O.TRANSACTION**

Directly met by FDP_ROL.1 and FDP_RIP.1 (more precisely, as specified by FDP_RIP.1.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_ROL.1/FIREWALL, and FDP_RIP.1/OBJECTS).

Transactions are provided to *applet*s as Java Card class libraries.

8.2.1.1.13 **O.CIPHER**

This objective is directly related to FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4 and FCS_COP.1. Another important SFR is FPR_UNO.1; the observation of the cryptographic operations may be used to disclose the keys.

The associated security functions are not described herein. They are provided to *applets* as Java class libraries (see the class **`javacardx.crypto.Cipher`** and the package **`javacardx.security`**).

8.2.1.1.14 **O.PIN-MNGT**

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

8.2.1.1.15 **O.KEY-MNGT**

This relies on the same functional requirements as O.CIPHER (FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FCS_COP.1), plus FDP_RIP.1 (FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/OBJECTS), FPR_UNO.1 and FDP_SDI.2 as well.

8.2.1.1.16 **O.REMOTE**

The access to the TOE's internal data and the flow of information from the card to the *CAD* required by the *JCRMI* service is under control of the *JCRMI access control policy* (*FDP_ACC.2/JCRMI*, *FDP_ACF.1/JCRMI*) and the ***JCRMI*** *information flow* control policy (*FDP_IFC.1/JCRMI*, *FDP_IFF.1/JCRMI*). The functional requirements of the class ***FMT*** included in the group *RMIG* also contribute to meet this objective.

8.2.1.1.17 **O.OBJ-DELETION**

This objective specifies that deletion of objects is secure. The objective is met by the functional requirements *FDP_RIP.1/ODEL* and *FPT_FLS.1/ODEL*.

8.2.1.1.18 **O.SCP.IC**

This objective is met by the component FPT_PHP.3 (FPT_PHP.3/SCP).

8.2.1.1.19 **O.SCP.RECOVERY**

This objective is met by the components FPT_FLS.1 (FPT_FLS.1/SCP), FPT_RCV.3 (FPT_RCV.3/SCP) and FRU_FLT.1 (FRU_FLT.1/SCP). The components FPT_RCV.3 and FPT_RCV.4 are used to support the objective O.SCP.SUPPORT and O.SCP.RECOVERY to assist the TOE to recover in the event of a power failure. If the power fails or the card is withdrawn prematurely from the CAD the operation of the TOE may be interrupted leaving the TOE in an inconsistent state.

8.2.1.1.20 **O.SCP.SUPPORT**

This objective is met by the components FPT_SEP.1 (FPT_SEP.1/SCP) (no bypassing TSF), FPT_AMT.1 (FPT_AMT.1/SCP), FPT_RCV.3 (FPT_RCV.3/SCP), FPT_RCV.4 (FPT_RCV.4/SCP), FPT_RVM.1 (FPT_RVM.1/SCP), FMT_LIM.1 and FMT_LIM.2.

The components FPT_RCV.3 and FPT_RCV.4 are used to support the objective O.SCP.SUPPORT and O.SCP.RECOVERY to assist the TOE to recover in the event of a power failure. If the power fails or the card is withdrawn prematurely from the CAD the operation of the TOE may be interrupted leaving the TOE in an inconsistent state.

8.2.1.1.21 O.CARD-MANAGEMENT

The objective O.CARD-MANAGEMENT is met by the SFRs of the Card Management Group (§5.1.9) (FDP_ACC.1 (FDP_ACC.1/CMGR), FDP_ACF.1 (FDP_ACF.1/CMGR), FMT_MSA.1 (FMT_MSA.1/CMGR), FMT_MSA.3 (FMT_MSA.3/CMGR), FMT_SMR.1 (FMT_SMR.1/CMGR), FIA_UID.1 (FIA_UID.1/CMGR)).

8.2.1.1.22 O.RND

This objective is met by the component FCS_RND.1

8.2.1.1.23 O.SIDE_CHANNEL

This objective is met by the component FPT_EMSEC.1.

8.2.1.1.24 O.CHECK_INIT

This objective is met by the component FPT_TST.1.2.
To ensure the receipt of the correct TOE from the IC Manufacturer by the TOE Manufacturer, the TSF shall provide authorised users (for example the TOE Manufacturer) with the capability to verify the integrity of the TSF data and therefore ensure the receipt of the correct TOE from the IC Manufacturer by the TOE Manufacturer.

8.2.1.2 Security objectives for the environment

8.2.1.2.1 OE.NATIVE

In this configuration all the security objectives directly or indirectly depend on the behavior of the native code embedded on the card. This trusted native code is not subject to change during the lifetime of the card. The objective OE.NATIVE ensures that the environmental assumption A.NATIVE is upheld. O.NATIVE mainly relies on the environmental objectives OE.NATIVE and in the requirement of secure security attributes expressed by the component FMT_MSA.2.

8.2.1.2.2 OE.APPLET

The environmental objective *OE.APPLET* might be also satisfied by IT procedural means. The IT verification that a post-issuance loaded applet contains no native code could be carried out as a part of the verification of how well the *CAP* file is formed. This verification has been associated in the group *BCVG* (Chapter 5.3.1) to the requirement of secure security attributes, expressed by the component FMT_MSA.2 (see note at 5.3.1.13).

8.2.1.2.3 OE.VERIFICATION

The environmental objective OE.VERIFICATION, which is satisfied by IT procedural means, is met by the SFRs of the group Bytecode Verification (§5.1.3) (FDP_IFC.2 (FDP_IFC.2/BCV), FDP_IFF.2 (FDP_IFF.2/BCV), FMT_MSA.1 (FMT_MSA.1/BCV.1, FMT_MSA.1/BCV.2), FMT_MSA.2 (FMT_MSA.2/BCV), FMT_MSA.3 (FMT_MSA.3/BCV), FMT_SMR.1 (FMT_SMR.1/BCV)) and FMT_SMF.1 (FMT_SMF.1.1/BCV).

8.2.2 Rationale tables of security objectives and security requirements

	O.SID	O.OPERATE	O.RESOURCES	O.FIREWALL	O.PIN-MNGT	O.INSTALL	O.CHECK_INIT
FDP_ACC.2		X		X	X		
FDP_ACF.1		X		X	X		
FDP_IFC.1				X			
FDP_IFF.1				X			
FDP_ROL.1		X	X		X		
FMT_MSA.2				X			
FMT_MSA.3	X			X			
FMT_MSA.1	X			X			
FMT_SMF.1				X			
FMT_SMR.1			X	X			
FPT_SEP.1	X	X		X			
FDP_RIP.1					X		
FPT_RCV.3		X	X			X	
FDP_ITC.2	X	X		X		X	
FPT_FLS.1		X	X			X	
FRU_RSA.1			X				
FDP_SDI.2					X		
FPR_UNO.1					X		
FAU_ARP.1		X	X				
FPT_RVM.1	X	X	X	X			
FPT_TDC.1		X					
FIA_UID.2	X						
FPT_TST.1		X					X

	O.SID	O.OPERATE	O.RESOURCES	O.FIREWALL	O.PIN-MNGT	O.INSTALL	O.CHECK_INIT
FPT_AMT.1		X					
FMT_MTD.1	X		X	X			
FMT_MTD.3	X		X	X			
FIA_ATD.1	X	X					
FIA_USB.1	X	X					

Table 24 Rationale of security objectives and functional requirements for the TOE (part 1/4)

	O.REALLOCATION	O.SHRD_VAR_CONFID	O.LOAD	O.SHRD_VAR_INTEG	O.RND	O.SIDE_CHANNEL
FDP_IFC.1		X		X		
FDP_IFF.1		X	X	X		
FDP_RIP.1	X	X				
FDP_IFC.2			X			
FIA_UID.1			X			
FCS_RND.1					X	
FPT_EMSEC.1						X
FCO_NRO.2			X			
FTP_ITC.1			X			

Table 25 Rationale of security objectives and functional requirements for the TOE (part 2/4)

	O.NATIVE	O.ALARM	O.SCP.IC	O.DELETION	O.OBJ-DELETION	O.REMOTE	O.TRANSACTION
FPT_FLS.1		X		X	X		
FAU_ARP.1		X					
FPT_RVM.1	X						
FPT_PHP.3			X				
FDP_ACC.2				X		X	
FDP_ACF.1				X		X	
FDP_RIP.1				X	X		X
FDP_IFC.1						X	
FDP_IFF.1						X	

	O.NATIVE	O.ALARM	O.SCP.IC	O.DELETION	O.OBJ-DELETION	O.REMOTE	O.TRANSACTION
FMT_MSA.1				X		X	
FMT_MSA.3				X		X	
FMT_REV.1						X	
FMT_SMR.1				X		X	
FPT_RCV.3				X			
FDP_ROL.1							X

Table 26 Rationale of security objectives and functional requirements for the TOE (part 3/4)

	O.CIPHER	O.KEY-MNGT	O.SCP.RECOVERY	O.SCP.SUPPORT	O.SCP.IC	O.CARD-MANAGEMENT
FCS_CKM.1	X	X				
FCS_CKM.2	X	X				
FCS_CKM.3	X	X				
FCS_CKM.4	X	X				
FCS_COP.1	X	X				
FDP.RIP.1		X				
FDP_SDI.2		X				
FPR_UNO.1	X	X				
FPT_AMT.1				X		
FPT_FLS.1			X			
FRU_FLT.1			X			
FPT_PHP.3					X	
FPT_SEP.1				X		
FPT_RVM.1				X		
FPT_RCV.3			X	X		
FPT_RCV.4				X		
FDP_ACC.1						X
FDP_ACF.1						X
FMT_LIM.1				X		
FMT_LIM.2				X		
FMT_MSA.1						X
FMT_MSA.3						X

	O.CIPHER	O.KEY-MNGT	O.SCP.RECOVERY	O.SCP.SUPPORT	O.SCP.IC	O.CARD-MANAGEMENT
FMT_SMR.1						X
FIA_UID.1						X

Table 27 Rationale of security objectives and functional requirements for the TOE (part 4/4)

	OE.VERIFICATION	OE.APPLET	OE.NATIVE
FDP_IFC.2	X		
FDP_IFF.2	X		
FMT_MSA.1	X		
FMT_MSA.2	X	X	X
FMT_SMF.1	X		
FMT_MSA.3	X		
FMT_SMR.1	X		
FRU_RSA.1	X		

Table 28 Rationale of security objectives and functional requirements for the environment

8.2.3 EAL rationale

EAL4 allows a developer to attain a reasonably high assurance level without the need for highly specialized processes and practices. It corresponds to a white box analysis and it can be considered as a reasonable level that can be applied to an existing product line without undue expense and complexity.

8.2.4 EAL augmentations rationale

8.2.4.1 AVA_VLA.4 Highly resistant

As a result, it is imperative that the TOE vulnerabilities to be reviewed be drawn from a systematic search rather than strictly a manufacturer prepared identification list. Component ***AVA_VLA.3*** requires that such a systematic search for vulnerabilities be documented and presented. This provides a significant increase in the consideration of vulnerabilities over that provided by ***AVA_VLA.2***. There might be scenarios, for example if the TOE is intended to stay in a hostile environment for long periods of time, or if the applications are considered to be highly sensitive, that would justify a further augmentation by requiring the component ***AVA_VLA.4.*** This latter component dictates that the TOE must be shown to be resistant to penetration attacks performed by attackers possessing a high attack potential.

AVA_VLA.4 has the following dependencies:

- ***ADV_FSP.1*** Informal functional specification
- ***ADV_HLD.2*** Security enforcing high-level design
- ***ADV_IMP.1*** Subset of the implementation of the TSF
- ***ADV_LLD.1*** Descriptive low-level design
- ***AGD_ADM.1*** Administrator guidance
- ***AGD_USR.1*** User guidance

All of these are met or exceeded in the EAL4 assurance package.

8.2.4.2 ADV_IMP.2 Implementation of the TSF

The implementation representation is used to express the notion of the least abstract representation of the TSF, specifically the one that is used to create the TSF itself without further design refinement.

The assurance component ***ADV_IMP.2*** has been chosen because the evaluation of the TOE must ensure that its security functional requirements are completely and accurately addressed by the implementation representation of the TSF.

ADV_IMP.2 has the following dependencies:

- ***ADV_LLD.1*** Descriptive low-level design
- ***ADV_RCR.1*** Informal correspondence demonstration
- ***ALC_TAT.1*** Well-defined development tools

All of these are met or exceeded in the EAL4 assurance package.

8.2.5 Security functional requirements dependencies

Requirements	**CC Dependencies**	**Satisfied Dependencies**
FAU_ARP.1/JCS	(FAU_SAA.1)	Unsupported, see: 8.2.5.1.1
FCO_NRO.2/CM	(FIA_UID.1)	FIA_UID.1/CM
FCS_CKM.1	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4) and (FMT_MSA.2)	FMT_MSA.2/JCRE, FCS_CKM.2, FCS_CKM.4
FCS_CKM.2	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	FMT_MSA.2/JCRE, FCS_CKM.1, FCS_CKM.4, FDP_ITC.2

Requirements	**CC Dependencies**	**Satisfied Dependencies**
	and (FCS_CKM.4) and (FMT_MSA.2)	
FCS_CKM.3	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4) and (FMT_MSA.2)	FMT_MSA.2/JCRE, FCS_CKM.1, FCS_CKM.4, FDP_ITC.2
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FMT_MSA.2)	FMT_MSA.2/JCRE, FCS_CKM.1, FDP_ITC.2
FCS_COP.1	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4) and (FMT_MSA.2)	FMT_MSA.2/JCRE, FCS_CKM.1, FCS_CKM.4, FDP_ITC.2
FCS_EMSEC.1	No dependencies	
FCS_RND.1	No dependencies	
FDP_ACC.1/CMGR	(FDP_ACF.1)	FDP_ACF.1/CMGR
FDP_ACC.1/ADEL	(FDP_ACF.1)	FDP_ACF.1/ADEL
FDP_ACC.2/FIREWALL	(FDP_ACF.1)	FDP_ACF.1/FIREWALL
FDP_ACF.1/CMGR	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/CMGR, FMT_MSA.3/CMGR
FDP_ACF.1/FIREWALL	(FDP_ACC.1) and (FMT_MSA.3)	FMT_MSA.3/FIREWALL, FDP_ACC.2/FIREWALL
FDP_ACC.2.1/JCRMI	(FDP_ACF.1)	FDP_ACF.1/JCRMI
FDP_ACF.1/ADEL	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/ADEL, FMT_MSA.3/ADEL
FDP_ACF.1/JCRMI	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.2.1/JCRMI, FMT_MSA.3/JCRMI
FDP_IFC.1/JCRMI	(FDP_IFF.1)	FDP_IFF.1/JCRMI
FDP_IFC.1/JCVM	(FDP_IFF.1)	FDP_IFF.1/JCVM
FDP_IFC.2/BCV	(FDP_IFF.1)	FDP_IFF.2/BCV
FDP_IFF.1/CM	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.1/CM, FMT_MSA.3/CM
FDP_IFF.1/JCRMI	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.1/JCRMI, FMT_MSA.3/JCRMI
FDP_IFF.1/JCVM	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.1/JCVM FMT_MSA.3/FIREWALL
FDP_IFF.2/BCV	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/BCV, FMT_MSA.3/BCV

Requirements	**CC Dependencies**	**Satisfied Dependencies**
FDP_ITC.2	(FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1) and (FPT_TDC.1)	OK: FPT_TDC.1, FDP_IFC.1/CM, FTP_ITC.1/CM
FDP_RIP.1/ABORT	No dependencies	
FDP_RIP.1/APDU	No dependencies	
FDP_RIP.1/bArray	No dependencies	
FDP_RIP.1/KEYS	No dependencies	
FDP_RIP.1/OBJECTS	No dependencies	
FDP_RIP.1/TRANSIENT	No dependencies	
FDP_ROL.1/FIREWALL	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.2/FIREWALL FDP_IFC.1/JCVM
FDP_SDI.2	No dependencies	
FDP_UIT.1/CM	(FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.1/CM, FTP_ITC.1/CM
FIA_ATD.1/AID	No dependencies	
FIA_UID.1/CM	None	
FIA_UID.1/CMGR	No dependencies	
FIA_UID.2/AID	No dependencies	
FIA_USB.1	(FIA_ATD.1)	FIA_ATD.1/AID
FMT_LIM.1	FMT_LIM.2	FMT_LIM.2
FMT_LIM.2	FMT_LIM.1	FMT_LIM.1
FMT_MSA.1/ ADEL	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMR.1)	FDP_ACC.1/ADEL, FMT_SMR.1/ADEL
FMT_MSA.1/BCV.1	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMR.1) and (FMT_SMF.1)	FMT_SMR.1/BCV, FDP_IFC.2/BCV, FMT_SMF.1.1/BCV
FMT_MSA.1/BCV.2	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMR.1) and (FMT_SMF.1)	FMT_SMR.1/BCV, FDP_IFC.2/BCV, FMT_SMF.1.1/BCV
FMT_MSA.1/CM	(FDP_ACC.1 or	FDP_IFC.1/CM,

Requirements	**CC Dependencies**	**Satisfied Dependencies**
	FDP_IFC.1) and (FMT_SMR.1)	FMT_SMR.1/CM
FMT_MSA.1/CMGR	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMR.1) and (FMT_SMF.1)	FDP_ACC.1/CMGR, FMT_SMR.1/CMGR, FMT_SMF.1.1/CMGR
FMT_MSA.1/EXPORT FMT_MSA.1/JCRMI FMT_MSA.1/REM-REFS	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMR.1)	FDP_IFC.1/JCRMI, FMT_SMR.1/JCRMI
FMT_MSA.1/JCRE	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMR.1) and (FMT_SMF.1)	FDP_IFC.1/JCVM, FMT_SMR.1/JCRE, FDP_ACC.2/FIREWALL, FMT_SMF.1.1/JCRE
FMT_MSA.2/BCV	(ADV_SPM.1) and (FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/BCV.1, FMT_SMR.1/BCV, FDP_IFC.2/BCV, ADV_SPM.1
FMT_MSA.3/ADEL	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/ADEL, FMT_SMR.1/ADEL
FMT_MSA.2/JCRE	(ADV_SPM.1) and (FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	FDP_IFC.1/JCVM, FMT_MSA.1/JCRE, FMT_SMR.1/JCRE, FDP_ACC.2/FIREWALL, ADV_SPM.1
FMT_MSA.3/BCV	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/BCV.1, FMT_SMR.1/BCV
FMT_MSA.3/CM	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/CM, FMT_SMR.1/CM
FMT_MSA.3/CMGR	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/CMGR, FMT_SMR.1/CMGR
FMT_MSA.3/FIREWALL	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/JCRE, FMT_SMR.1/JCRE
FMT_MSA.3/JCRMI	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/JCRMI, FMT_SMR.1/JCRMI
FMT_MTD.1/JCRE	(FMT_SMR.1) and (FMT_SMF.1)	FMT_SMR.1/JCRE, FMT_SMF.1.1/JCRE
FMT_MTD.3	(ADV_SPM.1) and (FMT_MTD.1)	FMT_MTD.1/JCRE, ADV_SPM.1

Requirements	**CC Dependencies**	**Satisfied Dependencies**
FMT_REV.1/JCRMI	(FMT_SMR.1)	FMT_SMR.1/JCRMI
FMT_SMF.1.1/JCRE	No dependencies	
FMT_SMF.1.1/BCV	No dependencies	
FMT_SMR.1/CM	(FIA_UID.1)	FIA_UID.1/CM
FMT_SMF.1.1/CMGR	No dependencies	
FMT_SMR.1/ADEL	(FIA_UID.1)	Unsupported, see: 8.2.5.1.4
FMT_SMR.1/BCV	(FIA_UID.1)	Unsupported, see: 8.2.5.1.2
FMT_SMR.1/CMGR	(FIA_UID.1)	FIA_UID.1/CMGR
FMT_SMR.1/JCRE	(FIA_UID.1)	FIA_UID.2/AID
FMT_SMR.1/Installer	(FIA_UID.1)	Unsupported, see: 8.2.5.1.3
FMT_SMR.1/JCRMI	(FIA_UID.1)	FIA_UID.1/AID
FPR_UNO.1	No dependencies	
FPT_AMT.1/SCP	No dependencies	
FPT_FLS.1/ADEL	(ADV_SPM.1)	ADV_SPM.1
FPT_FLS.1/Installer	(ADV_SPM.1)	ADV_SPM.1
FPT_FLS.1/JCS	(ADV_SPM.1)	ADV_SPM.1
FPT_FLS.1/ODEL	(ADV_SPM.1)	
FPT_FLS.1/SCP	(ADV_SPM.1)	ADV_SPM.1
FPT_RCV.3/Installer	(FPT_TST.1) and (AGD_ADM.1) and (ADV_SPM.1)	FPT_TST.1
FPT_PHP.3/SCP	No dependencies	
FPT_RCV.3/SCP	(ADV_SPM.1) and (AGD_ADM.1) and (FPT_TST.1)	FPT_TST.1, ADV_SPM.1, AGD_ADM.1
FPT_RCV.4/SCP	(ADV_SPM.1)	ADV_SPM.1
FPT_RVM.1	No dependencies	
FPT_RVM.1/SCP	No dependencies	
FPT_SEP.1	No dependencies	
FPT_SEP.1/SCP	No dependencies	

Requirements	CC Dependencies	Satisfied Dependencies
FPT_TDC.1	No dependencies	
FPT_TST.1	(FPT_AMT.1)	FPT_AMT.1/SCP
FRU_FLT.1/SCP	(FPT_FLS.1)	FPT_FLS.1/SCP
FRU_RSA.1/Installer	No dependencies	
FRU_RSA.1/BCV	No dependencies	
FTP_ITC.1/CM	No dependencies	

Table 29 Functional requirements dependencies

8.2.5.1 Rationale for the exclusion of dependencies

8.2.5.1.1 The dependency FAU_SAA.1 of FAU_ARP.1/JCS 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 TSP, 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[14].

8.2.5.1.2 The dependency FIA_UID.1 of FMT_SMR.1/BCV is unsupported.

This is required by the component FMT_SMR.1 in group BCVG. However, the role bytecode verifier defined in this component is attached to an IT security function rather than to a “user” of the CC terminology. The bytecode verifier does not “identify” itself with respect to the TOE; furthermore, it is part of the IT environment. Thus, here it is claimed that this dependency can be left out.

8.2.5.1.3 The dependency FIA_UID.1 of FMT_SMR.1/Installer is unsupported.

This is required by the component ***FMT_SMR.1*** in group *InstG*. However, the role installer defined in this component is attached to an IT security function rather than to a “user” of the CC terminology. The installer does not “identify” itself with respect to the

[14] No set of auditable events can be defined for the TOE.

TOE, but is a part of it. Thus, here it is claimed that this dependency can be left out. The reader may notice that the role is required because of the SFRs on management of TSF data and security attributes, essentially those of the firewall policy.

8.2.5.1.4 **The dependency FIA_UID.1 of FMT_SMR.1/ADEL is unsupported.**

This is also required by the component ***FMT_SMR.1*** in group ADELG. See the explanation in the paragraph above (the role in this case is applet deletion manager).

8.2.6 Security assurance requirements dependencies

Requirements	CC Dependencies	Satisfied Dependencies
AVA_VLA.4	(ADV_FSP.1) and (ADV_HLD.2) and (ADV_IMP.1) and (ADV_LLD.1) and (AGD_ADM.1) and (AGD_USR.1)	ADV_IMP.2, ADV_FSP.2, ADV_HLD.2, ADV_LLD.1, AGD_ADM.1, AGD_USR.1
ADV_IMP.2	(ADV_LLD.1) and (ADV_RCR.1) and (ALC_TAT.1)	ADV_LLD.1, ADV_RCR.1, ALC_TAT.1
ACM_AUT.1	(ACM_CAP.3)	ACM_CAP.4
ACM_CAP.4	(ALC_DVS.1)	ALC_DVS.1
ACM_SCP.2	(ACM_CAP.3)	ACM_CAP.4
ADO_DEL.2	(ACM_CAP.3)	ACM_CAP.4
ADO_IGS.1	(AGD_ADM.1)	AGD_ADM.1
ADV_FSP.2	(ADV_RCR.1)	ADV_RCR.1
ADV_HLD.2	(ADV_FSP.1) and (ADV_RCR.1)	ADV_FSP.2, ADV_RCR.1
ADV_LLD.1	(ADV_HLD.2) and (ADV_RCR.1)	ADV_HLD.2, ADV_RCR.1
ADV_RCR.1	No dependencies	
ADV_SPM.1	(ADV_FSP.1)	ADV_FSP.2
AGD_ADM.1	(ADV_FSP.1)	ADV_FSP.2
AGD_USR.1	(ADV_FSP.1)	ADV_FSP.2
ALC_DVS.1	No dependencies	
ALC_LCD.1	No dependencies	
ALC_TAT.1	(ADV_IMP.1)	ADV_IMP.2
ATE_COV.2	(ADV_FSP.1) and (ATE_FUN.1)	ADV_FSP.2, ATE_FUN.1
ATE_DPT.1	(ADV_HLD.1) and (ATE_FUN.1)	ADV_HLD.2, ATE_FUN.1
ATE_FUN.1	No dependencies	
ATE_IND.2	(ADV_FSP.1) and (AGD_ADM.1) and (AGD_USR.1) and (ATE_FUN.1)	ADV_FSP.2, AGD_ADM.1, AGD_USR.1, ATE_FUN.1
AVA_MSU.2	(ADO_IGS.1) and (ADV_FSP.1) and (AGD_ADM.1) and (AGD_USR.1)	ADO_IGS.1, ADV_FSP.2, AGD_ADM.1, AGD_USR.1

Requirements	CC Dependencies	Satisfied Dependencies
AVA_SOF.1	(ADV_FSP.1) and (ADV_HLD.1)	ADV_FSP.2, ADV_HLD.2

Table 30 Assurance requirements dependencies

8.2.7 Rationale for the strength of function

The TOE is intended to operate in open environments, where attackers can easily exploit vulnerabilities. The TOE also offers Java Card technology and GlobalPlatform 2.1.1 services to applets on the chip such as financial applets. According to the claimed intended usage of the TOE especially for fraud sensitive banking applets, it is very likely that it may represent a significant value and then constitute an attractive target for attacks. In some malicious usages of the TOE the statistical or probabilistic mechanisms in the TOE, for instance, may be subjected to analysis and attack in the normal course of operation.

A strength of function level high seems to be the reasonable level for cards hosting sensitive banking applications. Thus, in this security target, a protection against high attack potential has been chosen as the minimal level for those multi-application cards. For the following Security Functions an SOF-claim is appropriate as permutational but not cryptographical mechanisms are involved: SF.CRYPTO, SF.INTEGRITY.

For all these TSF the claim is SOF-high, which is appropriate to meet the requirements for resistance against attackers with high attack potential.

The strength of function level high is consistent with the vulnerability analysis level that has been specified (AVA_VLA.4).

8.3 TOE summary specification rationale

8.3.1 TOE security functions rationale

8.3.1.1 TOE security functional requirements

The following table gives the coverage of the TOE Security Functional Requirements by the TOE Security Functions. The numbers in the table give the corresponding component of the Security Function covering the requirement; the identified components obviously satisfy the requirements. Additional justifications discus the correspondence in more detail in chapter 8.3.1.3 in some cases.

8.3.1.2 Rationale table of functional requirements and security functions

	SF.TRANSACTION	SF.ACCESS_CONTROL	SF.CRYPTO	SF.INTEGRITY	SF.SECURITY	SF.APPLET	SF.RMI	SF.CARRIER	SF.CARDMANAGER
FAU_ARP.1.1/JCS					1[15]				
FCS_CKM.1.1			1, 4						
FCS_CKM.2.1			2						
FCS_CKM.3.1			2						
FCS_CKM.4.1			3						
FCS_COP.1.1			1, 4						
FCS_COP.1.1/SCP			4,5						
FDP_ACC.2.1/ FIREWALL,		1, 5							
FDP_ACF.1.1/FIREWALL, FDP_ACF.1.2/ FIREWALL		1							
FDP_IFC.1.1/JCVM		1							
FDP_IFF.1.1/JCVM, FDP_IFF.1.2/JCVM, FDP_IFF.1.3/JCVM, FDP_IFF.1.4/JCVM, FDP_IFF.1.5/JCVM, FDP_IFF.1.6/JCVM		1							
FDP_RIP.1.1/ABORT					1, 3				
FDP_RIP.1.1/APDU					1				
FDP_RIP.1.1/bArray					1				
FDP_RIP.1.1/KEYS					1				
FDP_RIP.1.1/OBJECTS					1				
FDP_RIP.1.1/TRANSIENT					1				
FDP_ROL.1.1/FIREWA	1, 2	1							

[15] The numbers in the table give the corresponding component of the Security Function covering the requirement.

	SF.TRANSACTION	SF.ACCESS_CONTROL	SF.CRYPTO	SF.INTEGRITY	SF.SECURITY	SF.APPLET	SF.RMI	SF.CARRIER	SF.CARDMANAGER
LL, FDP_ROL.1.2/FIREWALL									
FDP_SDI.2.1, FDP_SDI.2.2				1, 2, 3					
FIA_ATD.1.1/AID		2							
FIA_UID.2.1/AID		3							
FIA_USB.1.1		2, 3							
FIA_USB.1.2		1, 2, 4							
FIA_USB.1.3		6							
FMT_MSA.1.1/JCRE		6							
FMT_MSA.2.1/JCRE		4							
FMT_MSA.3.1/FIREWALL, FMT_MSA.3.2/FIREWALL		2							
FMT_MTD.1.1/JCRE		2							
FMT_MTD.3.1		4							
FMT_SMF.1.1/JCRE		2, 6							
FMT_SMR.1.1/JCRE, FMT_SMR.1.2/JCRE		1							
FPR_UNO.1.1					2				
FPT_FLS.1.1/JCS					1				
FPT_FLS.1.1/SCP					1				
FPT_RVM.1.1		1, 2, 3							
FPT_RVM.1.1/SCP		1, 2, 3							
FPT_SEP.1.1, FPT_SEP.1.2		1							
FPT_SEP.1.1/SCP,		1							

	SF.TRANSACTION	SF.ACCESS_CONTROL	SF.CRYPTO	SF.INTEGRITY	SF.SECURITY	SF.APPLET	SF.RMI	SF.CARRIER	SF.CARDMANAGER
FPT_SEP.1.2/SCP									
FPT_TDC.1.1, FPT_TDC.1.2		7							
FPT_TST.1.1				4					
FPT_TST.1.2, FPT_TST.1.3				5					
FDP_ITC.2.1/Installer, FDP_ITC.2.2/Installer, FDP_ITC.2.3/Installer, FDP_ITC.2.4/Installer, FDP_ITC.2.5/Installer						1 2 3 3 3			
FMT_SMR.1.1/Installer, FMT_SMR.1.2/Installer						1 1			
FPT_FLS.1.1/Installer						4			
FPT_RCV.3.1/Installer, FPT_RCV.3.2/Installer, FPT_RCV.3.3/Installer, FPT_RCV.3.4/Installer,						4 4 4 4			
FPT_RCV.3.1/SCP, FPT_RCV.3.2/SCP, FPT_RCV.3.3/SCP, FPT_RCV.3.4/SCP, FPT_RCV.4.1/SCP					1 1 1 1 1				
FRU_RSA.1.1/Installer						5			
FDP_ACC.2.1/ADEL, FDP_ACC.2.2/ADEL						6 6			
FDP_ACF.1.1/ADEL, FDP_ACF.1.2/ADEL, FDP_ACF.1.3/ADEL, FDP_ACF.1.4/ADEL						6 6 6 6			

	SF.TRANSACTION	SF.ACCESS_CONTROL	SF.CRYPTO	SF.INTEGRITY	SF.SECURITY	SF.APPLET	SF.RMI	SF.CARRIER	SF.CARDMANAGER
FMT_MSA.1.1/ADEL, FMT_MSA.3.1/ADEL, FMT_MSA.3.2/ADEL						7 7			
FMT_SMR.1.1/ADEL, FMT_SMR.1.2/ADEL						9 9			
FDP_RIP.1.1/ADEL						1			
FDP_RIP.1.1/ODEL						1, 8			
FPT_FLS.1.1/ADEL						9			
FPT_FLS.1.1/ODEL						9			
FPT_AMT.1.1/SCP				4					
FRU_FLT.1.1/SCP	1, 2		5	3					
FPT_PHP.3.1/SCP					1				
FDP_ACC.2.1/JCRMI FDP_ACC.2.2/JCRMI							1		
FDP_ACF.1.1/JCRMI FDP_ACF.1.2/JCRMI FDP_ACF.1.3/JCRMI FDP_ACF.1.4/JCRMI							1		
FDP_IFC.1.1/JCRMI FDP_IFF.1.1/JCRMI FDP_IFF.1.2/JCRMI FDP_IFF.1.3/JCRMI FDP_IFF.1.4/JCRMI FDP_IFF.1.5/JCRMI FDP_IFF.1.6/JCRMI							2		
FMT_MSA.1.1/JCRMI FMT_MSA.1.1/EXPORT FMT_MSA.1.1/REM_REFS							3		

	SF.TRANSACTION	SF.ACCESS_CONTROL	SF.CRYPTO	SF.INTEGRITY	SF.SECURITY	SF.APPLET	SF.RMI	SF.CARRIER	SF.CARDMANAGER
FMT_MSA.3.1/JCRMI FMT_MSA.3.2/JCRMI							3		
FMT_REV.1.1/JCRMI							4		
FMT_REV.1.2/JCRMI							5		
FMT_SMR.1.1/JCRMI FMT_SMR.1.2/JCRMI							6		
FCO_NRO.2.1/CM								1	
FCO_NRO.2.2/CM								2	
FCO_NRO.2.3/CM								3	
FIA_UID.1.1/CM								4	
FIA_UID.1.2/CM								5	
FDP_IFC.2.1/CM FDP_IFC.2.2/CM								6	
FDP_IFF.1.1/CM FDP_IFF.1.2/CM FDP_IFF.1.3/CM FDP_IFF.1.4/CM FDP_IFF.1.5/CM FDP_IFF.1.6/CM								6	
FDP_UIT.1.1/CM FDP_UIT.1.2/CM								6	
FMT_MSA.1.1/CM								6	
FMT_MSA.3.1/CM FMT_MSA.3.2/CM								7	
FMT_SMR.1.1/CM FMT_SMR.1.2/CM								8	
FTP_ITC.1.1/CM								9	
FTP_ITC.1.2/CM								10	
FTP_ITC.1.3/CM								11	

	SF.TRANSACTION	SF.ACCESS_CONTROL	SF.CRYPTO	SF.INTEGRITY	SF.SECURITY	SF.APPLET	SF.RMI	SF.CARRIER	SF.CARDMANAGER
FCS_RND.1.1			6						
FPT_EMSEC.1.1					4				
FPT_EMSEC.1.2					5				
FMT_LIM.1.1					5				
FMT_LIM.2.1					5				
FDP_ACC.1.1/CMGR, FDP_ACF.1.1/CMGR, FDP_ACF.1.2/CMGR, FDP_ACF.1.3/CMGR, FDP_ACF.1.4/CMGR,									1
FMT_MSA.1.1/CMGR									2
FMT_SMF.1.1/CMGR									3
FMT_MSA.3.1/CMGR, FMT_MSA.3.2/CMGR									4
FMT_SMR.1.1/CMGR, FMT_SMR.1.2/CMGR									5
FIA_UID.1.1/CMGR									6
FIA_UID.1.2/CMGR									7

Table 31 Functional requirements and security functions rationale

8.3.1.3 Justifications for the correspondence between functional requirements and security functions

FCS_CKM.1.1: The TOE generates cryptographic keys according to three different standards. The cryptographic algorithm provided by the JCAPI must than also be initialised.

FCS_COP.1.1: The cryptographic algorithms provided by the TOE have to be initialised.

FDP_ACC.2.1/ FIREWALL, FDP_ACF.1.1/FIREWALL: The FIREWALL access control SFP also defines the exceptions that are thrown if they are violated.

FIA_USB.1.1: Because the TOE requires each user to identify itself before allowing any other TSF-mediated actions on behalf of that user and restricts the ability to modify the list of registered applets and packages AID to the JCRE the TSF associate the user security attributes Package AID or “JCRE” with subjects acting on behalf of that user.

FPT_RVM.1.1: If the TOE maintains a security domain for its own execution and requires each user to identify itself before allowing any other TSF-mediated actions on behalf of that user and restricts the ability to modify the list of registered applets and packages AID to the JCRE than the TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

FRU_FLT.1.1/SCP: If the TOE runs out of EEPROM, the optimistic backup includes a backup of the previous data value at first data modification, and previous value restoring at abort. Additionally the TOE still provides cryptographic operations. However, upon detection of a data integrity error on keys or the associated security attributes during an EDC test the TOE will throw an exception to secure the operations of memory management and cryptographic algorithms.

FCS_RND.1.1: The TOE generates random numbers according to standards with strength of function high.

FPT_EMSEC.1.1/FPT_EMSEC.1.2: The TOE does not emit useful information and prevents gaining access to useful information by the electrical contacts interface.

FMT_LIM.1.1/FMT_LIM.2.1: Depoying test features after TOE delivery does not allow the manipulation or disclosure of useful information.

FDP_ACC.1.1/CMGR, FDP_ACF.1.1/CMGR, FDP_ACF.1.2/CMGR, FDP_ACF.1.3/CMGR, FDP_ACF.1.4/CMGR: The CARD CONTENT MANAGEMENT access control SFP on S.CAD, S.CARDMANAGER, S.CRD, O.PACKAGE and O.APPLET defines rules to determine if an operation among controlled subjects and controlled objects is allowed.

FMT_MSA.1.1/CMGR: The CARD CONTENT MANAGEMENT access control SFP restricts the ability to modify the security attributesard Life Cycle State, Security Level to the card manager.

FMT_SMF.1.1/CMGR: Security attributes Card Life Cycle State and Security Level can be modified by the TOE.

FMT_MSA.3.1/CMGR, FMT_MSA.3.2/CMGR: The CARD CONTENT MANAGEMENT access control SFP to provide restrictive default values for security attributes that are used to enforce the SFP.

FMT_SMR.1.1/CMGR, FMT_SMR.1.2/CMGR: The TOE maintains the roles S.CAD, S.CRD and S.CARDMANAGER.

FIA_UID.1.1/CMGR: The TOE allows GET DATA, INITIALIZE UPDATE and EXTERNAL AUTHENTICATE to be performed before the user is identified.

FIA_UID.1.2/CMGR: The TOE requires S.CAD, S.CRD and S.CARDMANAGER to be successfully identified before allowing any other TSF-mediated actions.

8.3.2 Assurance measures rationale

8.3.2.1 TOE security assurance requirements

ACM Configuration management

ACM_AUT CM automation

8.3.2.1.1 **ACM_AUT.1** The configuration management is described in SMARTCAFE_ACM.

ACM_CAP CM capabilities

8.3.2.1.2 **ACM_CAP.4** The configuration management is described in SMARTCAFE_ACM.

ACM_SCP CM scope

8.3.2.1.3 **ACM_SCP.2** The configuration management is described in SMARTCAFE_ACM.

ADO Delivery and operation

ADO_DEL Delivery

8.3.2.1.4 **ADO_DEL.2** The delivery, installation, generation and start-up of the TOE is described in SMARTCAFE_ADO.

ADO_IGS Installation, generation and start-up

8.3.2.1.5 **ADO_IGS.1** The delivery, installation, generation and start-up of the TOE is described in SMARTCAFE_ADO.

ADV Development

ADV_FSP Functional specification

8.3.2.1.6 **ADV_FSP.2** The representation of the TSF is described in SMARTCAFE_ADV_SPM for security policy modelling, in SMARTCAFE_ADV_FSP for functional specification, in SMARTCAFE_ADV_HLD for high level design, in SMARTCAFE_ADV_LLD for low level design, in SMARTCAFE_ADV_IMP for implementation representation and in SMARTCAFE_ADV_RCR for representation correspondence.

ADV_HLD High-level design

8.3.2.1.7 **ADV_HLD.2** The representation of the TSF is described in SMARTCAFE_ADV_SPM for security policy modelling, in SMARTCAFE_ADV_FSP for functional specification, in SMARTCAFE_ADV_HLD for high level design, in SMARTCAFE_ADV_LLD for low level design, in SMARTCAFE_ADV_IMP for implementation representation and in SMARTCAFE_ADV_RCR for representation correspondence.

ADV_LLD Low-level design

8.3.2.1.8 **ADV_LLD.1** The representation of the TSF is described in SMARTCAFE_ADV_SPM for security policy modelling, in SMARTCAFE_ADV_FSP for functional specification, in SMARTCAFE_ADV_HLD for high level design, in SMARTCAFE_ADV_LLD for low level design, in SMARTCAFE_ADV_IMP for implementation representation and in SMARTCAFE_ADV_RCR for representation correspondence.

ADV_RCR Representation correspondence

8.3.2.1.9 **ADV_RCR.1** The representation of the TSF is described in SMARTCAFE_ADV_SPM for security policy modelling, in SMARTCAFE_ADV_FSP for functional specification, in SMARTCAFE_ADV_HLD for high level design, in SMARTCAFE_ADV_LLD for low level design, in SMARTCAFE_ADV_IMP for implementation representation and in SMARTCAFE_ADV_RCR for representation correspondence.

ADV_SPM Security policy modelling

8.3.2.1.10 **ADV_SPM.1** The representation of the TSF is described in SMARTCAFE_ADV_SPM for security policy modelling, in SMARTCAFE_ADV_FSP for functional specification, in SMARTCAFE_ADV_HLD for high level design, in SMARTCAFE_ADV_LLD for low level design, in SMARTCAFE_ADV_IMP for implementation representation and in SMARTCAFE_ADV_RCR for representation correspondence.

AGD Guidance documents

AGD_ADM Administrator guidance

8.3.2.1.11 **AGD_ADM.1** The guidance documentation is described in SMARTCAFE_AGD_USR for the user and in SMARTCAFE_AGD_ADM for the administrator.

AGD_USR User guidance

8.3.2.1.12 **AGD_USR.1** The guidance documentation is described in SMARTCAFE_AGD_USR for the user and in SMARTCAFE_AGD_ADM for the administrator.

ALC Life cycle support

ALC_DVS Development security

8.3.2.1.13 **ALC_DVS.1** The life cycle support of the TOE during its development and maintenance is described in SMARTCAFE_ALC.

ALC_LCD Life cycle definition

8.3.2.1.14 **ALC_LCD.1** The life cycle support of the TOE during its development and maintenance is described in SMARTCAFE_ALC.

ALC_TAT Tools and techniques

8.3.2.1.15 **ALC_TAT.1** The life cycle support of the TOE during its development and maintenance is described in SMARTCAFE_ALC.

ATE Tests

ATE_COV Coverage

8.3.2.1.16 **ATE_COV.2** The testing of the TOE is described in SMARTCAFE_ATE.

ATE_DPT Depth

8.3.2.1.17 **ATE_DPT.1** The testing of the TOE is described in SMARTCAFE_ATE.

ATE_FUN Functional tests

8.3.2.1.18 **ATE_FUN.1** The testing of the TOE is described in SMARTCAFE_ATE.

AVA Vulnerability assessment

AVA_MSU Misuse

8.3.2.1.19 **AVA_MSU.2** The vulnerability assessment for the TOE is described in SMARTCAFE_AVA_MSU for the misuse, in SMARTCAFE_AVA_SOF for the strength of TOE security functions and in SMARTCAFE_AVA_VLA for the vulnerability analysis.

AVA_SOF Strength of TOE security functions

8.3.2.1.20 **AVA_SOF.1** The vulnerability assessment for the TOE is described in SMARTCAFE_AVA_MSU for the misuse, in SMARTCAFE_AVA_SOF for the strength of TOE security functions and in SMARTCAFE_AVA_VLA for the vulnerability analysis.

Miscellaneous

8.3.2.1.21 **AVA_VLA.4** The vulnerability assessment for the TOE is described in SMARTCAFE_AVA_MSU for the misuse, in SMARTCAFE_AVA_SOF for the strength of TOE security functions and in SMARTCAFE_AVA_VLA for the vulnerability analysis.

8.3.2.1.22 **ADV_IMP.2** The representation of the TSF is described in SMARTCAFE_ADV_SPM for security policy modelling, in SMARTCAFE_ADV_FSP for functional specification, in SMARTCAFE_ADV_HLD for high level design, in SMARTCAFE_ADV_LLD for low level design, in SMARTCAFE_ADV_IMP for implementation representation and in SMARTCAFE_ADV_RCR for representation correspondence.

8.3.2.2 Rationale table of assurance requirements and assurance measures

	AM_ACM	AM_ADO	AM_ADV	AM_AGD	AM_ALC	AM_ATE	AM_AVA
ACM_AUT.1	X						
ACM_CAP.4	X						
ACM_SCP.2	X						
ADO_DEL.2		X					
ADO_IGS.1		X					
ADV_FSP.2			X				
ADV_HLD.2			X				
ADV_LLD.1			X				
ADV_RCR.1			X				
ADV_SPM.1			X				
AGD_ADM.1				X			
AGD_USR.1				X			
ALC_DVS.1					X		
ALC_LCD.1					X		
ALC_TAT.1					X		
ATE_COV.2						X	
ATE_DPT.1						X	
ATE_FUN.1						X	
ATE_IND.2						X	
AVA_MSU.2							X
AVA_SOF.1							X

	AM_ACM	AM_ADO	AM_ADV	AM_AGD	AM_ALC	AM_ATE	AM_AVA
AVA_VLA.4							X
ADV_IMP.2			X				

Table 32 Assurance requirements and assurance measures rationale

8.4 Definition of additional Families

This chapter has been taken from the certified Common Criteria Protection Profile — Machine Readable Travel Document with ICAO Application, Extended Access Control (PP-MRTD EAC), Version 1.1, 07.09.2006, BSI-PP-0026 [EAC].

8.4.1 Definition of Family FCS_RND

To define the IT security functional requirements of the TOE an additional family (FCS_RND) of the Class FCS (cryptographic support) is defined in [EAC]. This family describes the functional requirements for random number generation used for cryptographic purposes. The component FCS_RND is not limited to generation of cryptographic keys as the component FCS_CKM.1 is. The similar component FIA_SOS.2 is intended for non-cryptographic use.

The family “Generation of random numbers (FCS_RND)” is specified as follows.

8.4.1.1 FCS_RND Generation of random numbers

Family behaviour

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

Component levelling:

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

Management: FCS_RND.1

There are no management activities foreseen.

Audit: FCS_RND.1

There are no actions defined to be auditable.

FCS_RND.1 Quality metric for random numbers

Hierarchical to: No other components.

8.4.1.1.1 FCS_RND.1.1

The TSF shall provide a mechanism to generate random numbers that meet [assignment: *a defined quality metric*].

Dependencies: No dependencies.

8.4.2 Definition of the Family FPT_EMSEC

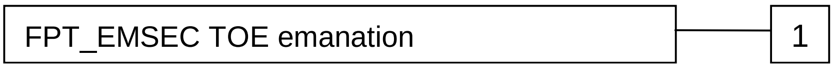
The additional family FPT_EMSEC (TOE Emanation) of the Class FPT (Protection of the TSF) is defined in [EAC] to describe the IT security functional requirements of the TOE. The TOE shall prevent attacks against the logical MRTD data and other secret data where the attack is based on external observable 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 [2].

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

Family behaviour

This family defines requirements to mitigate intelligible emanations.

Component levelling:



FPT_EMSEC.1 TOE emanation has two constituents:

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

FPT_EMSEC.1.2 Interface Emanation requires not emit interface emanation enabling access to TSF data or user data.

Management: FPT_EMSEC.1

There are no management activities foreseen.

Audit: FPT_EMSEC.1

There are no actions defined to be auditable.

8.4.2.1 FPT_EMSEC.1 TOE Emanation

Hierarchical to: No other components.

8.4.2.1.1 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*].

8.4.2.1.2 FPT_EMSEC.1.2

The TSF shall ensure [assignment: *type of users*] are unable to use the following interface [assignment: *type of connection*] to gain access to [assignment: *list of types of TSF data*] and [assignment: *list of types of user data*].

Dependencies: No other components.

8.4.3 Definition of the Family FMT_LIM

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

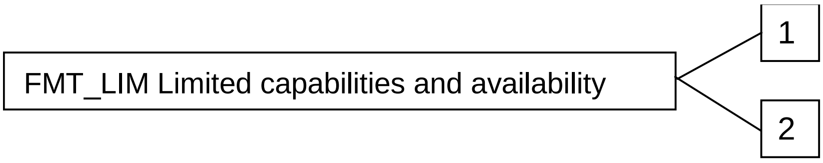
The family “Limited capabilities and availability (FMT_LIM)” is specified as follows.

8.4.3.1 FMT_LIM Limited capabilities and availability

Family behaviour

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

Component levelling:



FMT_LIM.1 Limited capabilities requires that the TSF is built to provide only the capabilities (perform action, gather information) necessary for its genuine purpose.

FMT_LIM.2 Limited availability requires that the TSF restrict the use of functions (refer to Limited capabilities (FMT_LIM.1)). This can be achieved, for instance, by removing or by disabling functions in a specific phase of the TOE’s life cycle.

Management: FMT_LIM.1, FMT_LIM.2

There are no management activities foreseen.

Audit: FMT_LIM.1, FMT_LIM.2

There are no actions defined to be auditable.

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

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

8.4.3.1.1 FMT_LIM.1 Limited capabilities

Hierarchical to: No other components.

FMT_LIM.1.1 The TSF shall be designed in a manner that limits their capabilities so that in conjunction with “Limited availability (FMT_LIM.2)” the following policy is enforced [assignment: *Limited capability and availability policy*].

Dependencies: FMT_LIM.2 Limited availability.

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

8.4.3.1.2 FMT_LIM.2 Limited availability

Hierarchical to: No other components.

FMT_LIM.2.1 The TSF shall be designed in a manner that limits their availability so that in conjunction with "Limited capabilities (FMT_LIM.1)" the following policy is enforced [assignment: *Limited capability and availability policy*].

Dependencies: FMT_LIM.1 Limited capabilities.

8.5 Statement of compatibility

This is a statement of compatibility between this Composite Security Target (Composite-ST) and the Platform Security Targets (Platform-STs) of the NXP Chip P5CD040/80/144 [STLCD40], [STLCD80], [STLCD144]. This statement is compliant to the requirements of [SUPP].

8.5.1 Classification of Platform TSFs

A classification of TSFs of the Platform-ST has been made. Each TSF has been classified as 'relevant' or 'not relevant' for the Composite-ST.

TOE Security Functions	**Relevant**	**Not relevant**
F.RNG: Random Number Generator	x	
F.HW_DES: Triple-DES Co-processor	x	
F.HW_AES: AES Co-processor	x	
F.OPC: Control of Operating Conditions	x	
F: PHY: Protection against Physical Manipulation	x	
F.LOG: Logical Protection	x	
F.COMP: Protection of Mode Control	x	

TOE Security Functions	Relevant	Not relevant
F.MEM_ACC: Memory Access Control		x
F.SFR_ACC: Special Function Register Access Control		x

Table 33 Classification of Platform-TSFs

F.MEM_ACC and F.SFR_ACC are not relevant of the Composite-ST because

the combination of F.SFR_ACC and F.COMP ensures that the other CPU modes are not available for the Smartcard Embedded Software, but reserved for specific purposes fulfilled by the IC Dedicated Software.

All other listed TSFs of the Platform-ST are relevant for the Composite-ST.

8.5.2 Matching statement

The TOE needs the following implicit assumptions of the IC (see 2.2):

- Certified NXP Microcontroller P5CD040/80/144
- True Random Number Generator (TRNG) with P2 SOF-high classification according to AIS 31 [AIS31]
- Cryptographic support based on asymmetric and symmetric key algorithms (RSA, 3TDES) with 2048 bit asymmetric key length and 168 bit symmetric cryptographic key length.

8.5.2.1 TOE Security Environment

8.5.2.1.1 Threats

(see chapter 3.5.)

The following threats of this Composite-ST are directly related to IC functionality:

- T.PHYSICAL
- T.RND
- T.LEAKAGE

These threats will be mapped to the following Platform-ST threats and OSPs:

- T.Leak-Inherent
- T.Phys_Probing
- T.Phys_Manipulation
- T.Malfunction

- T.Abuse_Func
- T.Leak-Forced
- T.RND
- P.Add-Components (Additional Specific Security Components)
- P.Process-TOE (Protection during TOE Development and Production)

The table below shows the mapping of the threats.

	Threats/OSPs									
		P.Add-Components	P.Process-TOE	T.Leak-Inherent	T.RND	T.Phys_Probing	T.Phys_Manipulation	T.Malfunction	T.Abuse_Func	T.Leak-Forced
Composite-ST	T.PHYSICAL	X	X			X	X	X	X	
	T.RND				X					
	T.LEAKAGE	X	X	X						X

Table 34 Mapping of threats

The threats from chapter 3.5 not mentioned here are not related to the Platform-ST.

T.PHYSICAL matches to T.Phys_Probing, T.Phys_Manipulation, T.Malfunction, T.Abuse_Func as all these threats are directed against the SCP in a direct or indirect physical way. Additionaly P.Add_components and P.Process-TOE as a specific security components policy matches with T.PHYSICAL and T.LEAKAGE because it prevents modification of configuration data during TOE Development and Production and after TOE delivery e.g.

T.RND matches to T.RND which is trivial.

T.LEAKAGE matches to T.Leak-Inherent and T.Leak-Forced and also P.Add_components and P.Process-TOE because all threats and OSP are dealing also with leakage threats.

8.5.2.1.2 Assumptions

(see chapter 3.4)

The assumptions from this ST (A.NATIVE, A.VERIFICATION and A. APPLETT) make no assumption to the Platform, but to processes and users.

The assumption from the Platform-ST:

Assumptions of Platform-ST	**Claasification of significant assumptions**	**Mapping to Security Objectives of this Composite-ST**
A.Process-Card	IrPA (irrelevant platform assumption)	n/a
A.Plat-Appl	IrPA	n/a
A.Resp-Appl	CfPA (automatically fulfilled platform assumption)	O.SIDE_CHANNEL
A.Check-Init	CfPA	O.CHECK_INIT
A.Key-Function	CfPA	O.KEY_MNGT, O.SCP.IC, O.SIDE_CHANNEL

Table 35 Mapping of assumptions

There is no significant platform assumption (SgPA) of the Platform-ST fore this Composite-ST.

There is **no conflict** between **security environments** of this Composite-ST and the Platform-ST [[STLCD40], [STLCD80], [STLCD144]].

8.5.2.1.3 Security objctives

Security objectives see: chapter 4

This Composite-ST has security objectives which are directly related to the Platform-ST.

These are:

- O.SCP.RECOVERY
- O.SCP.SUPPORT
- O.SCP.IC
- O.SIDE_CHANNEL

These objectives will be mapped to the following Platform-ST objectives:

- O.Leak-Inherent
- O.Phys-Probing
- O.Malfunction
- O.Phys-Manipulation
- O.Leak-Forced
- O.Abuse-Func
- O.RND

The mapping is shown below.

		Platform-ST						
		O.Leak-Inherent	O.RND	O.Phys-Probing	O.Malfunction	O.Phys-Manipulation	O.Leak-Forced	O.Abuse-Func
Composite-ST	Obejctives for TOE_IC							
	O.SCP.RECOVERY				X			
	O.SCP.SUPPORT	X	X	X	X	X	X	X
	O.SCP.IC	X	X	X	X	X	X	X
	O.RND		X					
	O.SIDE_CHANNEL	X		X		X	X	X

Table 36 Mapping of objectives

O.SCP.RECOVERY matches to O.Malfunction because they allow the TOE to eventually complete the interrupted operation successfully.

O.SCP.SUPPORT matches all listed objectives of the Platform-ST because they provide functionality that support the well-functioning of the TSFs of the TOE (avoiding they are bypassed or altered) and by controlling the access to information proper of the TSFs

O.SCP.IC matches to all listed objectives of the Platform-ST because they all describe IC security features.

O.RND matches to O.RND which is trivial.

O.SIDE_CHANNEL matches to O.Leak-Inherent, O.Phys-Probing, O.Phys-Manipulation, O.Leak-Forced and O.Abuse-Func because they provide protection against disclosure of primary assets including confidential data (User Data or TSF data) stored and/or processed in the Smart Card IC to avoid interpretations of signals extracted from the hardware part of the TOE (Power Supply, Electro Magnetic emissions, e.g.).

Other objectives of this Composite-ST in chapter 4.1 are not applicable for the mapping to the Platform-ST. These objectives are:

OT.AC_Pers

OT.Additional_Applications

OT.Services

Only OT.Cryptography, which requires specification conformant cryptographic algorithms, has a link to the Platform-ST as the classification of the TRNG is defined in the specification as well. But the TRNG described in the Platform-ST matches the requirements of the specification.

The Objectives for the Operational Environment (see 4.2) are all not linked to the platform and are therefore not applicable to this mapping.

There is **no conflict** between **security objectives** of this Composite-ST and the Platform-ST [STLCD40], [STLCD80], [STLCD144].

8.5.2.1.4 Security requirements

Security Functional Requirements see chapter 5.1

This Composite-ST has the following platform related SFRs:

FPT_EMSEC.1

FPT_FLS.1

FPT_PHP.3

FPT_TST.1

FMT_LIM.1

FPT_AMT.1.1/SCP

FPT_RCV.3.1/SCP

FCS_RND.1.1

FCS_COP.1.1

These SFRs will be matched by the following SFRs of the Platform-ST:

FPT_FLS.1

FRU_FLT.2

FDP_ITT.1

FPT_ITT.1

FPT_PHP.3

FCS_RND.1

FCS_COP.1 [DES]

FCS_COP.1 [AES]

The matching will be as described in the table below.

		Platform-ST							
		FPT_FLS.1	FCS_RND.1	FCS_COP.1[DES]	FCS_COP.1[AES]	FRU_FLT.2	FDP_ITT.1	FPT_ITT.1	FPT_PHP.3
Composite-ST									
	FPT_EMSEC.1						X	X	
	FPT_FLS.1/SCP	X				X			X
	FPT_PHP.3/SCP					X			X
	FPT_TST.1					X			
	FMT_LIM.1					X			
	FPT_AMT.1.1/SCP					X			
	FPT_RCV.3.1/SCP	X				X			X
	FCS_RND.1.1		X						
	FCS_COP.1.1			X	X				

Table 37 Mapping of SFRs

FPT_EMSEC.1 matches to FPT_ITT.1 and FDP_ITT.1 as the Composite-TOE should not emit information about IC power consumption and execution time while the Platform protects user and TSF data by leackage attacks.

FPT_FLS.1 and FPT_RCV.3.1/SCP matches to FPT_FLS.1, FRU_FLT.2 and FPT_PHP.3 as the Composite-TOE should preserve a secure state when the Platform operates out of normal operating conditions, while the Platform should ensure the robustness and operate always in a secure state.

FPT_PHP.3 matches the robustness requirements of FRU_FLT.2 and FPT_PHP.3.

FPT_TST.1, FMT_LIM.1 and FPT_AMT.1.1/SCP matches FRU_FLT.2 as the Platform must ensure that the TOE operates correctly within some limits.

FCS_RND.1.1 matches FCS_RND.1 as the Platform provides a mechanism to generate random numbers for the composite product and *to* provide entropy of at least 7.976 bit in each byte.

FCS_COP.1.1 matches FCS_COP.1 [DES] and FCS_COP.1 [AES] as the Platform provides encryption and decryprion for the composite product with the necessary cryptographic key sizes.
RSA cryptograhivs operations are supported by the ST-Platform through the FameXE co-processor through basic arithmetic functions for large integer numbers.

8.5.2.2 Assurance requirements

The Composite-ST requires EAL 4 augmented by:

ADV_IMP.2

AVA_VLA.4

The Platform-ST requires EAL 5 augmented by:

ALC_DVS.2

AVA_MSU.3

AVA_VLA.4

As EAL 5 covers all assurance requirements of EAL 4 all non augmented parts of the Composite-ST will match to the Platform-ST assurance requirements. But also the augmented parts of the Composite-ST match to the Platform-ST:

ADV_IMP.2 is part of EAL 5 and AVA_VLA.4 is augmented for the Platform-ST as well.

8.5.3 Overall no contracdictions found

Overall there is **no conflict** between **security requirements** of this Composite-ST and the Platform-ST [STLCD40], [STLCD80], [STLCD144].

9 References and Acronyms

9.1 References

[AES] *Federal Information Processing Standards Publication 197, ADVANCED ENCRYPTION STANDARD (AES)*, November 26, 2001

[AIS20] *Anwendungshinweise und Interpretationen zum Schema (AIS), AIS 20; Bundesamt für Sicherheit in der Informationstechnik*, Version 1.0, 2.12.1999

[AIS31] *Anwendungshinweise und Interpretationen zum Schema, AIS31: Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren*, Version 3.1, 25.09.2001, Bundesamt für Sicherheit in der Informationstechnik.

[CC1] *Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model*. Version 2.3. August 2005. CCIMB-2005-08-001.

[CC2] *Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements*. Version 2.3. August 2005. CCIMB-2005-08-002.

[CC3] *Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements*. Version 2.3. August 2005. CCIMB-2005-08-003.

[CCFI_003] *Common Criteria Final Interpretation 003, Unique identification of configuration items in the configuration list*. 11 February 2002.

[CCFI_004] *Common Criteria Final Interpretation 004, ACM_SCP.*.1C requirements unclear*. 12 November 2001.

[CCFI_051] *Common Criteria Final Interpretation 051, Use of documentation without C & P elements*. 25 October 2002.

[CCFI_056] *Common Criteria Final Interpretation 056*, When can the FPT_RCV dependency on FPT_TST be argued away?, 31 October 2003.

[CCFI_065] *Common Criteria Final Interpretation 065, No component to call out security function management*, 31. July 2001.

[CCFI_103] *Common Criteria Final Interpretation 103,* Association of access control attributes with subjects and objects, 15 July 2003.

[CCFI_104] *Common Criteria Final Interpretation 104,* Association of information flow attributes with subjects and objects, 15 July 2003.

[CCFI_137] *Common Criteria Final Interpretation 137, Rules governing binding should be specifiable.* 30 January 2004.

[CEM] *Common Methodology for Information Technology Security Evaluation, Evaluation Methodology*. Version 2.3, August 2005. CCMB-2005-08-004.

[DES] *National Institute of Standards and Technology, Data Encryption Standard, Federal Information Processing Standards Publication 46-3, October 25, 1999.*

[EAC] Common Criteria Protection Profile — Machine Readable Travel Document with ICAO

Application, Extended Access Control (PP-MRTD EAC), Version 1.1, 07.09.2006, BSI-PP-0026.

[ISO] *ISO/IEC 7816-3 : Second edition 1997-09-18, Identification cards - Integrated circuit(s) cards with contacts - Part 3 : Electronic signals and transmission protocols, ISO/IEC FCD 7816-4: 2003 (Draft) Identification cards - Integrated circuit(s) cards with contacts, Part 4: Interindustry commands for interchange, Working draft dated 2003-01-17, ISO SC17 Document 17N2268T, ISO/IEC 7816-5 : 1994, Identification cards - Integrated circuit(s) cards with contacts - Part 5 : Numbering system and registration procedure for application identifiers, ISO/IEC FCD 7816-6 : 2003 (Draft), Identification cards - Integrated circuit(s) cards with contacts - Part 6 : Interindustry data elements for interchange – FCD dated 2003-01-17, ISO SC17 Document 17N2270T, ISO/IEC FCD 7816-8: 2003 (Draft), Integrated circuit(s) cards with contacts, Part 8: Interindustry commands for a cryptographic toolbox. FCD dated 2003-01-17, ISO SC17 Document 17N2272T, ISO/IEC FCD 7816-9: 2003 (Draft), Integrated circuit(s) cards with contacts, Part 9: Interindustry commands for card and file management. FCD dated 2003- 01-17, SC17 Document 17N2274T.*

[ISO9797] ISO/IEC 9797-1:1999: Information technology – Security techniques –Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher

[JCVM22] *Java Card 2.2.2 Virtual Machine (JCVM) Specification*. March 2006. Published by Sun Microsystems, Inc.

[JCAPI22] *Java Card 2.2.2 Application Programming Interface*. March 2006. Published by Sun Microsystems, Inc.

[JCRE22] *Java Card 2.2.2 Runtime Environment (JCRE) Specification*. March 2006. Published by Sun Microsystems, Inc.

[JCBV] *Java Card 2.2 Off-Card Verifier*. June 2002. White paper. Published by Sun Microsystems, Inc.

[JCSPP] *Java Card System Protection Profile Collection Version 1.0b, Standard 2.2 Configuration, DCSI PP 0305, Sun Microsystems Inc., August 2003*

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

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

[NXP40] Certification Report BSI-DSZ-CC-0404-2007 for NXP Secure Smart Card Controller P5CD040V0B, P5CD020V0B, P5CC021V0B and P5CC040V0B each with specific IC Dedicated Software from NXP Semiconductors Germany GmbH Business Line Identification, Bundesamt für Sicherheit in der Informationstechnik (BSI), 05.07.07

[NXP80] Certification Report BSI-DSZ-CC-0410-2007 for NXP Secure Smart Card Controller P5CD080V0B, P5CN080V0B and P5CC080V0B each with specific IC Dedicated Software from NXP Semiconductors Germany GmbH Business Line Identification, Bundesamt für Sicherheit in der Informationstechnik (BSI), 05.07.07

[NXP144] Certification Report BSI-DSZ-CC-0411-2007 for NXP Secure Smart Card Controller P5CD144V0B, P5CN144V0B and P5CC144V0B each with specific IC Dedicated Software from NXP Semiconductors Germany GmbH Business Line Identification, Bundesamt für Sicherheit in der Informationstechnik (BSI), 05.07.07

[PKCS1] PKCS #1: RSA Encryption Standard – An RSA Laboratories Technical Note, Version 1.5, Revised November 1, 1993

[PP0010] *Protection Profile Smart Card IC with Multi-Application Secure Platform*. Version 2.0, Issue November 2000. Registered and Certified by the French Certification Body under the reference PP/0010.

[RSA] ***RFC 3447,*** J. Jonsson, B. Kaliski, ***"Public-Key Cryptography Standards (PKCS) #1: RSA*** Cryptography Specifications Version 2.1", February 2003

[SCPP] Smartcard IC Platform Protection Profile, Version 1.0, July 2001 (BSI-PP-0002)

[STLCD40] Security Target Lite, NXP P5CD040 P5CC040 P5CD020/ P5CC021 V0B, Rev. 1.1, 09.May 2007; BSI-DSZ-CC-0404

[STLCD80] Security Target Lite, NXP P5CD080/P5CN080/P5CC080 V0B, Rev. 1.1, 09.May 2007; BSI-DSZ-CC-0410

[STLCD144] Security Target Lite, NXP P5CD144/P5CN144/P5CC144 V0B, Rev. 1.0, 21.March 2007; BSI-DSZ-CC-0411

[SUPP] Supporting Document, Mandatory Technical Document, Composite product evaluation for Smart Cards and similar devices, September 2007, Version 1.0, CCDB-007-09-001.

[VISA] Visa GlobalPlatform 2.1.1 Card Implementation Requirements, Version 2.0, July 2007

9.2 Acronyms

CBC Cipher Block Chaining

CC Common Criteria

CCFI Common Criteria Final Interpretation

DS Dedicated software

EAL Evaluation Assurance Level

ECB Electronic Code Book

IT Information Technology

PP Protection Profile

SCP Smart Card Platform. It is comprised of the integrated circuit, the operating system and the dedicated software of the smart card.

SF Security Function

SFP Security Function Policy

SOF Strength of Function

ST Security Target

TOE Target of Evaluation

TSC TSF Scope of Control

TSF TOE Security Functions

TSFI TSF Interface

TSP TOE Security Policy

10 Appendix: Glossary

AID

Application identifier, an ISO-7816 data format used for unique identification of Java Card applications (and certain kinds of files in card file systems). The Java Card platform uses the *AID* data format to *identify* **applet**s *and packages*. *AIDs* are administered by the International Standards Organization (ISO), so they can be used as unique identifiers.

AIDs are also used in the security policies (see "*Context*" below): applets' *AIDs* are related to the selection mechanisms, *packages*' *AIDs* are used in the enforcement of the *firewall*. **Note:** although they serve different purposes, they share the same name space.

APDU

Application Protocol Data Unit, an ISO 7816-4 defined communication format between the card and the off-card applications. Cards receive requests for service from the CAD in the form of *APDU*s. These are encapsulated in Java Card System by the **`javacard.framework.APDU`** *class* ([JCAPI22]).

*APDU*s manage both the selection-cycle of the *applets* (through *JCRE* mediation) and the communication with the *Currently selected applet*.

APDU buffer

The APDU buffer is the buffer where the messages sent (received) by the card depart from (arrive to). The *JCRE* owns an *APDU* object (which is a *JCRE Entry Point* and an instance of the **`javacard.framework.APDU`** *class*) that encapsulates *APDU* messages in an internal byte array, called the *APDU buffer*. This object is made accessible to the *Currently selected applet* when needed, but any permanent access (out-of selection-scope) is strictly prohibited for security reasons.

applet

The name given to a Java Card technology-based user application. An applet is the basic piece of code that can be selected for execution from outside the card. Each applet on the card is uniquely identified by its *AID*.

applet deletion manager

The on-card component that embodies the mechanisms necessary to delete an applet or library and its associated data on smart cards using Java Card technology.

BCV

The bytecode verifier is the software component performing a static analysis of the code to be loaded on the card. It checks several kinds of properties, like the correct format of *CAP file*s and the enforcement of the typing rules associated to bytecodes. If the component is placed outside the card, in a secure environment, then it is called an off-card verifier. If the component is part of the embedded software of the card it is called an on-card verifier.

CAD

Card Acceptance Device or card reader. The device where the card is inserted, and which is used to communicate with the card.

CAP file

A file in the Converted applet format. A CAP file contains a binary representation of a *package* of *class*es that can be installed on a device and used to execute the *package*'s *class*es on a Java Card virtual machine. A CAP file can contain a user library, or the code of one or more applets.

Card tearing

An unexpected removal of the Card out of the CAD.

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 behaviour.

Each class declares a collection of fields and methods associated to its instances. The contents of the fields determine the internal state of a class instance, and the methods the operations that can be applied to it. Classes are ordered within a class hierarchy. A class declared as a specialization (a subclass) of another class (its super class) inherits all the fields and methods of the latter.

Java platform classes should not be confused with the classes of the functional requirements (FIA) defined in the CC.

Context

A context is an object-space partition associated to a *package*. Applets within the same Java technology-based *package* belong to the same context. The *firewall* is the boundary between contexts (see "*Current context*").

Current context

The *JCRE* keeps track of the current Java Card System context (also called "the active context"). When a virtual method is invoked on an object, and a context switch is required and permitted, the current context is changed to correspond to the context of the *applet* that owns the object. When that method returns, the previous context is restored. Invocations of static methods have no effect on the current context. The current context and sharing status of an object together determine if access to an object is permissible.

Currently selected applet

The applet has been selected for execution in the current session. The *JCRE* keeps track of the currently selected Java Card applet. Upon receiving a SELECT command from the *CAD* with this applet's *AID*, the *JCRE* makes this applet the currently selected applet. The *JCRE* sends all *APDU* commands to the currently selected applet (Glossary).

Default applet

The applet that is selected after a card reset ([JCRE22], §4.1).

Embedded Software

Pre-issuance loaded software.

Firewall

The mechanism in the Java Card technology for ensuring *applet* isolation and object sharing. The firewall prevents an applet in one *context* from unauthorized access to objects owned by the *JCRE* or by an applet in another context.

Installer

The installer is the on-card application responsible for the installation of applets on the card. It may perform (or delegate) mandatory security checks according to the card issuer policy, loads and link *package*s (*CAP file*(s)) on the card to a suitable form for the *JCVM* to execute the code they contain. It is a subsystem of what is usually called "card manager"; as such, it can be seen as the portion of the card manager that belongs to the TOE.

The installer has an *AID* that uniquely identifies him, and may be implemented as a Java Card applet. However, it is granted specific privileges on an implementation-specific manner ([JCRE22], §10).

Interface

A special kind of Java programming language *class*, which declares methods, but provides no implementation for them. A class may be declared as being the implementation of an interface, and in this case must contain an implementation for each of the methods declared by the interface. (see also *shareable interface*).

JCRE

The Java Card runtime environment consists of the Java Card virtual machine, the Java Card API, and its associated native methods. This notion concerns all those dynamic features that are specific to the execution of a Java program in a smart card, like *applet* lifetime, applet isolation and object sharing, transient objects, the transaction mechanism, and so on.

JCRE Entry Point

An object owned by the *JCRE* context but accessible by any application. These methods are the gateways through which applets request privileged *JCRE* system services: the instance methods associated to those objects may be invoked from any context, and when that occurs, a context switch to the *JCRE* context is performed.

There are two categories of JCRE Entry Point Objects: Temporary ones and Permanent ones. As part of the *firewall* functionality, the *JCRE* detects and restricts attempts to store references to these objects.

JCRMI

Java Card Remote Method Invocation is the Java Card System, version 2.2, mechanism enabling a client application running on the *CAD* platform to invoke a method on a remote object on the card. Notice that in Java Card System, version 2.1.1, the only method that may be invoked from the CAD is the **`process`** method of the **`applet`** class.

Java Card System

The Java Card System consists of the *JCRE* (*JCVM* +API). GlobalPlatform defines the Card Manager, which includes the *Installer* and the Applet Deletion Manager, which are also part of the TOE.

JCVM

The embedded interpreter of bytecodes. The JCVM is the component that enforces separation between applications (*firewall*) and enables secure data sharing.

logical channel

A logical link to an application on the card. A new feature of the Java Card System, version 2.2, that enables the opening of up to four simultaneous sessions with the card, one per logical channel. Commands issued to a specific logical channel are forwarded to the active applet on that logical channel.

Object deletion

The Java Card System, version 2.2, mechanism ensures that any unreferenced persistent (transient) object owned by the current context is deleted. The associated memory space is recovered for reuse prior to the next card reset.

Package

A *package* is a name space within the Java programming language that may contain *class*es and *interface*s. A *package* defines either a user library, or one or more applet definitions. A *package* is divided in two sets of files: export files (which exclusively contain the public *interface* information for an entire *package* of *class*es, for external linking purposes; export files are not used directly in a Java Card virtual machine) and *CAP file*s.

SCP

Smart Card Platform. It is comprised of the integrated circuit, the operating system and the dedicated software of the smart card.

Shareable interface

An interface declaring a collection of methods that an *applet* accepts to share with other applets. These *interface* methods can be invoked from an *applet* in a *context* different from the context of the object implementing the methods, thus "traversing" the *firewall*.

SIO

An object of a class implementing a *shareable interface*.

Subject

An active entity within the TOE that causes information to flow among objects or change the system's status. It usually acts on the behalf of a user. Objects can be active and thus are also *subjects* of the TOE.

Transient object

An object whose contents is not preserved across CAD sessions. The contents of these objects are cleared at the end of the current CAD session or when a card reset is performed. Writes to the fields of a transient object are not affected by transactions.

User	Any application interpretable by the *JCRE*. That also covers the *package*s. The associated subject(s), if applicable, is (are) an object(s) belonging to the **javacard.framework.applet** *class*.

End of Document