Security Target Lite
Sm@rtCafé® Expert 7.0 C3

Version 2.9/Status 16.08.2017

Author	Giesecke+Devrient Mobile Security GmbH /stut
Status	Final
Rating	Public
Edition	Version 2.9/Status 16.08.2017

Giesecke+Devrient Mobile Security GmbH
Prinzregentenstraße 159
D-81677 München

Contents

1 Introduction

1.1 ST Identification

Title: Security Target Lite Sm@rtCafé® Expert 7.0 C3
Reference: ASE_Sm@rtCafé® Expert 7.0 C3

Version Number: Version 2.9/Status 16.08.2017

Origin: Giesecke+Devrient Mobile Security GmbH

Author: stut/ MSRD34

Compliant to: Java Card Protection Profile, Open Configuration, May 2012, Version 3.0 [JCSPP].

TOE Reference: Sm@rtCafé® Expert 7.0 C3

TOE documentation:

- Preparative Guidance, [UGPre]
- Operative Guidance, [UGOpe]

HW-Part of TOE:

IFX M5073 G11 (Certificate: BSI-DSZ-CC-0951-2015, including re-assessment BSI-DSZ-CC-0951-2015-RA-01), [IFX_Cert], [IFX_STLite].

1.2 TOE Overview

This document is the Security Target for the TOE Sm@rtCafé® Expert 7.0 C3.

The Target of Evaluation (TOE) described in this ST is a dual-interface, contact based or a pure contactless smart card with a Javacard operating system (OS). The TOE is a multi-purpose Java card where applets of different kinds can be installed. Since a post-issuance installation of applets is possible, the TOE corresponds to an *open configuration*, as defined in [JCSPP]. Depending on the installed applets, the entire product (consisting of the TOE plus applets) can be used as a government card (like an ID card or a passport), a payment card, a signature card and other purposes.

The card is based on the Integrated Circuit (IC) [IFX_STLite] manufactured by IFX. It is a dual-interface, contact based or a pure contactless chip with maximum of 628 kBytes of flash memory. This hardware platform has been evaluated according to CC EAL6+ in compliance with the protection profile [PP0084]. The TOE is subject to a composite evaluation according to CC EAL5+.

The TOE mainly consists of the hardware mentioned above and the card OS Sm@rtCafé® Expert - a Javacard operating system based on the Javacard standards [JCVM304], [JCAPI3], [JCAPI304] and [JCRE304].

After mask development under the responsibility of G&D, the cards are delivered to the Composite Product Integrator (who might also be G&D).

1.3 Sections Overview

Section 1 provides the introductory material for the Security Target.

Section 2 provides general purpose and TOE description.

Section 3 contains the conformance claims for the TOE.

Section 4 defines the security aspects for TOE.

Section 5 contains the security problem definition.

Section 6 contains the security objectives for the TOE and its environment, including the security objectives rationale.

Section 8 contains the security functional requirements, including the security requirements rationale.

Section 9 contains the TOE summary specification.

Section 10 provides a statement of compatibility between the composite TOE and the hardware TOE.

Section 11 contains references, abbreviations and a glossary.

1.4 Typographic Conventions

- *This typeface* is used to highlight those words that appear in the Glossary. Example: *applet*.
- **<u>This typeface</u>** is used to highlight assignments, selections and refinements 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.

1.5 Change History

Version	Date	Changes	Responsible
2.9	16.08.17	Final Version	stut

1.6 Figures

1.7 Tables

1.8 Application Notes of the PP

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

2 TOE Description

2.1 TOE type

The TOE under evaluation is Sm@rtCafé® Expert 7.0 C3, a dual-interface, contact based or a pure contactless smart card with a Javacard operating system.

The Sm@rtCafé® Expert 7.0 C3 TOE consists of the following parts:

1. The smart card platform (SCP) consisting of the IC including its firmware and the OS. There is no IFX crypto library part of the TOE.
2. The native G&D crypto library is used by the implementation of some Java Card APIs (e.g. cryptographic libraries). This code cannot be used from the outside of the card directly.
3. The Java Card System (JCS) is implemented on top of the SCP. It is made up of the Java Card Runtime Environment (JCRE), the Java Card Virtual Machine (JCVM), the Java Card API, the on-card installer, the applet deletion manager and the smart card OS.
4. The Card Manager is the central administrator of the card.
5. The APDU Layer is used by an external Card Acceptance Device (CAD), i.e. an off card application, to send commands to the TOE and receive data from the TOE.
6. The TOE authentication keys (initial or customer specific):
   - SCP02- and SCP03-based authentication keys (for TOE configuration 1 and 2)

Applets are loaded on top of the JCS either pre- or post-issuance and are not part of the TOE.

An ST claiming conformance to [JCSPP] (see there in chapter 1.2) “shall comprehend the IC and all the embedded software, including the OS, the JCS, as well as additional native code and the pre-issuance applets”.

In addition to the TOE boundary defined in the PP, the Card Manager is also chosen as part of the TOE for this security target.

Therefore, the TOE boundary corresponds to the dotted line shown in Figure 1.

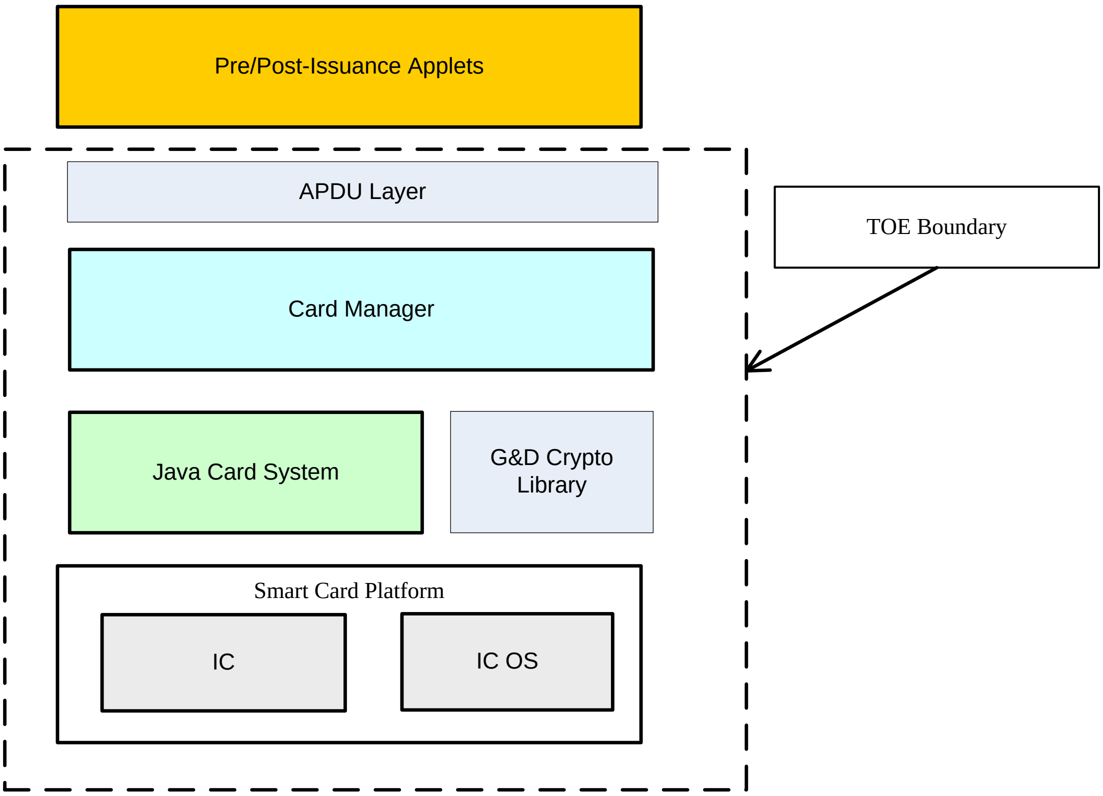

Figure 1: TOE boundary (dotted line)

The product containing the TOE is based on and designed to be compliant to the following specifications:

- The Java Card specification (see: [JCVM304], [JCRE304], [JCAPI304]);
- GlobalPlatform Card Common Implementation Specification [GP CIC].

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.

The TOE can be used in two different configurations:

- Config 1: TOE is compliant to the GlobalPlatform Card Common Implementation Configuration [GP CIC],
- Config 2: TOE is compliant to the GlobalPlatform Card ID Configuration [GP ID_Config].

All configurations can either be installed on a dual-interface, contact based chip or on a pure contactless smart card platform (SCP).

The User can identify the specific TOE configuration by the TOE response to a specific APDU specified in the Operative Guidance Sm@rtCafé® Expert 7.0 C3, [UGOpe].

2.2 Product Type

2.2.1 Physical scope of TOE

The TOE consists of the following parts:

- the hardware platform IFX M5073 G11 (Certificate: : BSI-DSZ-CC-0951-2015, including re-assessment BSI-DSZ-CC-0951-2015-RA-01), [IFX_Cert]) with the following configurations according to [IFX_STLite]:
  - FLASH: up to 628 kByte
  - ROM: not available
  - RAM for the user: 1-12 kByte
  - SCP (Symmetric Crypto Co-processor for DES and AES Standards): accessible
  - Crypto2304T (Crypto Co-processor for asymmetric algorithms like RSA and EC): accessible
  - Interfaces: ISO/IEC 7816 and/or ISO/IEC 14443
- Java Card Runtime Environment (JCRE)
- Java Card Virtual Machine (JCVM)
- Java Card API
- On-card Installer
- Applet Deletion Manager
- Card Manager
- Smart Card OS including the G&D crypto library

Java Card Remote Method Invocation (JCRMI) is <u>not</u> supported by the TOE.

2.2.2 Logical scope of TOE

The TOE provides the following services:

- Logical Channels
- Object Deletion
- Transaction and atomicity concept according to [JCRE304]
- firewall access control
- Cryptographic services by the G&D crypto library: RSA and ECC signature, RSA, DES and AES cipher/decipher, SHA hash algorithms, MAC, random number generation (for details see 8.1.1.2)
- enhanced G&D APIs (see chapter 4.2.3, Operative Guidance Sm@rtCafé® Expert 7.0 C3, [UGOpe])
- integrity check of checksum-protected data

- secure state of information
- non-observability of operations on sensitive information
- unavailability of previous information content
- secure installation of post-issuance applications on the card
- secure post-issuance deletion of previously installed applets

2.2.3 Non TOE Features of Sm@rtCafé® Expert 7.0 C3

The following SW-module of Sm@rtCafé® Expert 7.0 is not part of the TOE and can be part of the product or not:

• Biometric API

2.3 TOE environment

The following sections further describe the components involved in the environment of the Java Card System. The role they play will help in understanding the importance of the assumptions on the environment of the TOE.

2.3.1 Applet development

The development of applets is carried out in a Java programming environment. The compilation of the code produces the corresponding class file. Then all class files of the package are processed by the converter[1], which validates the code and generates a converted CAP-file, the equivalent of a Java™ package for the Java Card platform. A CAP file contains an executable binary representation of the classes of a package. A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of Java Card technology, it defines either a user library, or one or several applets.

2.3.2 Off-Card Verifier

The bytecode verifier is a program that performs static checks on the bytecodes of the methods of a CAP file prior to the execution of the file on the card.

2.3.3 Loading CAP files

After the validation is carried out, the CAP file is loaded into the card by means of a safe loading mechanism.

First an authentication step by which the card issuer and the card recognize each other by using a type of cryptographic certification (Secure Channel Protocol = 02, see [GP221] or Secure Channel Protocol = 03, see [GP AM D] and also see 2.3.3

[1] The converter is defined in the specifications [JCVM22] as the off-card component of the Java Card virtul machine.

and 9.1.7). Once the identification step is accomplished, the CAP file is transmitted to the card. Due to resource limitations, usually the file is split by the card issuer into a list of Application Protocol Data Units (APDUs), which are in turn sent to the card. Authentication of the external entity, loading and initialisation are parts of the TOE security features.

The Off-Card Loader is a program outside the smart card which transmits the executable binary in a CAP file to the On-Card Loader via a card reader.

The On-Card Loader (or installer) is a program inside the smart card which writes the binary received from the Off-Card Loader into the smart card memory.

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

The linking process consists of a rearrangement of the information contained in the CAP file in order to speed up the execution of the applications.

2.3.4 Components not belonging to the TOE environment

<u>On-Card Verifier</u>

The product does not contain an On-Card Verifier.

2.4 TOE life cycle

2.4.1 General life cycle

The TOE life cycle is part of the product life cycle, i.e. the Java Card platform with applications, which goes from product development to its usage by the final user. The product life cycle phases are those detailed in Figure 2. We refer to [PP0084] for a thorough description of Phases 1 to 7:

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

The TOE life cycle is composed of four stages:

- Development,
- Storage, pre-personalisation and testing
- Personalisation

- Final usage.

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

TOE development is performed during Phase 1. This includes JCS conception, design, implementation, testing and documentation. The JCS as part of the software development fulfils requirements of the final product, including conformance to Java Card Specifications, and recommendations of the SCP user guidance (reference see in [IFX_STLite]). The development occurs in a controlled environment that avoids disclosure of source code, data and any critical documentation and that guarantees the integrity of these elements. The software development environment is included in the evaluation of the TOE.

In Phase 3, the Security IC Manufacturer or the TOE developer himself will store, pre-personalize the TOE and potentially conduct tests on behalf of the TOE developer. The Security IC Manufacturing and the TOE developer environment protect the integrity and confidentiality of the TOE and of any related material, for instance test suites. The whole Security IC Manufacturing and TOE developer environment, in particular that location where the TOE is accessible for installation or testing is included in the evaluation of the TOE.

In this phase the FLASH loader is active. However, before the TOE can be delivered into phase 5 the FLASH loader has to be deactivated irreversible either by the Security IC Manufacturer or the TOE developer.

The TOE delivery takes place after Phase 4 so that the evaluation process is limited to Phases 1 to 4.

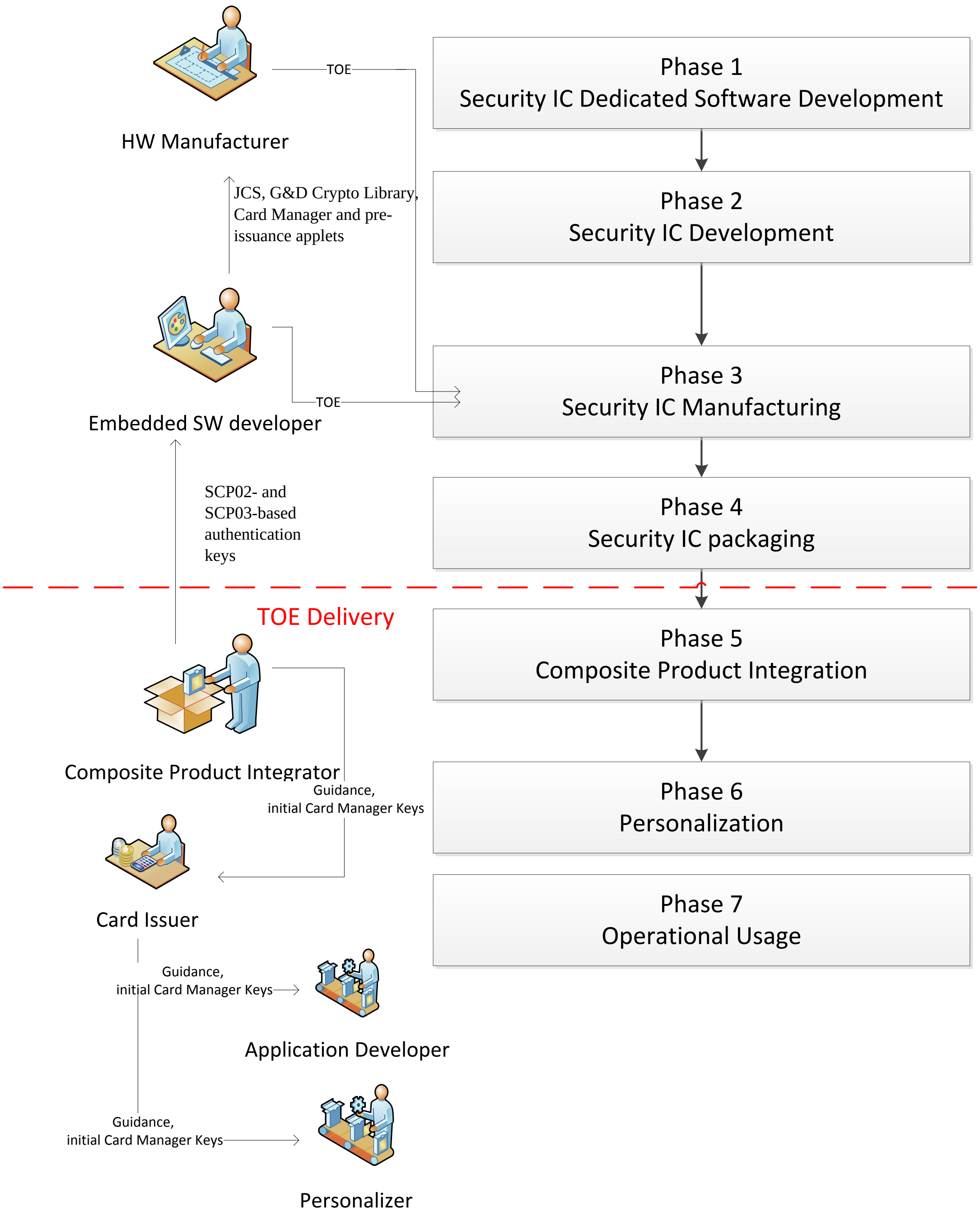

Figure 2: TOE Life Cycle within Product Life Cycle

In Phase 5, the Composite Product Integrator pre-personalizes the TOE (e.g. changes the card manager keys), installs pre-issuance applets on the EEPROM part of the TOE (only trustworthy applets shall be installed) and potentially conducts tests on behalf of the developer.

The Composite Product Integration environment protects the integrity and confidentiality of the TOE and of any related material, for instance test suites. The corresponding environment is not included in the product evaluation because the product delivery takes place after Phase 4.

The TOE is personalized in Phase 6. The Personalization environment is not included in the product evaluation.

The product shall be tested again and all critical material including personalization data, test suites and documentation shall be protected from disclosure and modification. The TOE final usage environment is that of the product where the JCS and the card manager is embedded. It covers a wide spectrum of situations that cannot be covered by evaluations.
However, only trustworthy applets shall be installed on the TOE.

In Phase 5 and 6 pre-issuance installation of applets into the EEPROM part of the TOE will take place.
Post-issuance installation of applets will take place in phase 7.

For the installation of applets technical and organizational measures associated to OE2 objective must be employed (see: [JIL] and the Operative and Preparative Guidance of Sm@rtCafé® Expert 7.0 C3, [UGOpe], [UGPre]).

The JCS, the card manager and the product shall provide the full set of security functionalities to avoid abuse of the product by untrusted entities.

2.4.2 Delivery scope of TOE

Delivery and acceptance procedures shall guarantee the authenticity, the confidentiality and integrity of the exchanged pieces. TOE delivery shall involve encrypted signed sending and it supposes the previous exchange of public keys. The delivery process is included in the evaluation of the TOE.

The Composite Product Integrator delivers the SCP02- and SCP03-based authentication keys (for TOE configuration 1 and 2) to the embedded SW developer.

The HW manufacturer may receive the software part of the TOE including JCS, G&D Crypto Library, Card Manager and pre-issuance applets (see figure 1) from the embedded SW developer and loads it on the Smart Card Platform (with or without the FLASH loader).

The HW manufacturer may also deliver the Smart Card Platform to the embedded SW developer who loads the software part of the TOE on the chips with the FLASH loader.

In both cases, TOE delivery takes place after phase 4: The parts of the TOE to be delivered are the ICC including the software part of the TOE. The embedded SW developer is responsible for the TOE delivery into phase 5.
Before the TOE can be delivered into phase 5 the FLASH loader has to be deactivated irreversible.
Besides the TOE the initial card manager key and the following documentation for the Smart Card issuer and applet developer are delivered:

- Preparative Guidance Sm@rtCafé® Expert 7.0 C3, [UGPre],

- Operative Guidance Sm@rtCafé® Expert 7.0 C3, [UGOpe].

The Composite Product Integrator delivers the Preparative Guidance Sm@rtCafé® Expert 7.0 C3, [UGPre] via the Card Issuer to the Application Developer and the personalizer and the Operative Guidance Sm@rtCafé® Expert 7.0 C3, [UGOpe] via the Card Issuer to the Application Developer.

The personalised TOE is delivered by the personalizer either directly or via the Composite Product Integrator to the end user. Only necessary guidance information is sent by the composite product integrator via the card issuer to the end user.

2.5 TOE usage

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

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

Applications installed on a Java Card platform can be selected for execution when the card communicates with a card reader.
Only trustworthy applets should be installed on the TOE.

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

So far, the most typical applications are:

- Financial applications, like Credit/Debit ones, stored value purse, or electronic commerce, among others.
- Transport and ticketing, granting pre-paid access to a transport system like the metro and bus lines of a city.
- Telephony, through the subscriber identification module (SIM) or the NFC chip for mobile phones.
- Personal identification, for granting access to secured sites or providing identification credentials to participants of an event.
- Electronic passports and identity cards.
- Secure information storage, like health records, or health insurance cards.
- Loyalty programs, like the “Frequent Flyer” points awarded by airlines.

3 Conformance Claims

3.1 CC conformance claims

This ST claims conformance to:

- Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, September 2012, version 3.1, revision 4, CCMB-2012-09-001 [CC1],
- Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements, September 2012, version 3.1, revision 4, CCMB-2012-09-002 [CC2],
- Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements, September 2012, version 3.1, revision 4, CCMB-2012-09-003 [CC3].

as follows

- Part 2 extended,
- Part 3 conformant.

3.2 Conformance claim to a PP

This ST claims *strict* conformance to the Protection Profile “Java Card Protection Profile - Open Configuration, May 2012, Version 3.0, Oracle Corporation” ([JCSPP]). The chosen PP configuration is 3 Classic Edition without optional feature external memory (EMG) and RMI (RMIG) as listed in table 1 of appendix 1 [JCSPP].

3.3 Conformance claim to a package

This ST claims conformance to:

Package EAL5 augmented with ALC_DVS.2 and AVA_VAN.5 components.

3.4 Conformance Claim Rationale

This security target is conformant to the claimed PP [JCSPP].

The TOE type described in chapter 2.1 is consistent with the “TOE of the ST” described in [JCSPP], chapter 2.1.2.

The Security Problem Definition (chapter 5) is taken directly from the PP ([JCSPP], chapter 5) with a few changes described therein.

The security requirements (chapter 8) have been taken directly from the PP ([JCSPP], chapter 7) and operations as appropriate have been performed.

3.5 PP additions and refinements

The following changes with respect to threats, assumptions, OSPs and objectives have been made:

1. The assumption “A.DELETION” has been replaced by a threat “T.SECURE_DELETION”.
2. T.EXE-CODE-REMOTE has been removed.
3. The security objective for the environment OE.CARD-MANAGEMENT is transformed into a security objective for the TOE O.CARD-MANAGEMENT.
4. The security objectives for the environment concerning the smart card platform (OE.SCP.IC, OE.SCP.RECOVERY and OE.SCP.SUPPORT) have been changed into objectives for the TOE (O.SCP.IC, O.SCP.RECOVERY and O.SCP.SUPPORT).
5. The objective O.REMOTE has been removed.
6. The following SFRs have been added:
   a. FTP_ITC.1/CMGR
   b. FPT_PHP.3
   c. FCS_RNG.1.1 (this family was additional refined)
   d. FPT_TST.1

For more detailed explanations see in the corresponding chapters.

4 Security Aspects

Chapter 4 of the PP [JCSPP] is adopted without changes.

5 Security Problem Definition

Chapter 5 of the PP [JCSPP] is adopted with 2 changes:

1. The assumption “A.DELETION” has been deleted, and instead a new threat “T.SECURE_DELETION” introduced because A.DELETION refers to the card manager which has been defined in chapter 2.1 to be part of the TOE (instead of being part of the TOE environment).
2. The threat T.EXE-CODE-REMOTE has been deleted because it refers to remote execution which is not supported by the TOE.

5.1 Additional Threat

The definition of the additional threat T.SECURE_DELETION is based on the security aspect #DELETION given in [JCSPP], chapter 4.5.

5.1.1 T.SECURE_DELETION

The attacker exploits security holes that are introduced through the deletion of an installed applet in the form of broken references to garbage collected code or data or alter integrity or confidentiality of remaining applets. That could be used to maliciously bypass the TSF and jeopardize the TOE (or its assets) in case of failure (such as power shortage).

Directly threated asset(s): D.APP_I_DATA, D.SEC_DATA, D_APP_KEY, D.PIN and D.CRYPTO.

6 Security objectives

Chapter 6 of the PP [JCSPP] has been basically adopted, with 3 changes:

1. The security objective for the environment OE.CARD-MANAGEMENT is transformed into a security objective for the TOE O.CARD-MANAGEMENT.
2. The security objectives for the environment concerning the smart card platform (OE.SCP.IC, OE.SCP.RECOVERY and OE.SCP.SUPPORT) have been changed into objectives for the TOE (O.SCP.IC, O.SCP.RECOVERY and O.SCP.SUPPORT) because the smart card platform has been defined to be part of the TOE.
3. The objective O.REMOTE has been removed because it refers to remote execution which is not supported by the TOE (see 2.2.1).

The text from chapter 6 of the PP has been copied, but for easier reading the changed parts have been underlined.

6.1 Security objectives for the TOE

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

6.1.1 IDENTIFICATION

O.SID

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

6.1.2 EXECUTION

O.FIREWALL

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

O.GLOBAL_ARRAYS_CONFID

The TOE shall ensure that the APDU buffer that is shared by all applications is always cleaned upon applet selection.

The TOE shall ensure that the global byte array used for the invocation of the install method of the selected applet is always cleaned after the return from the install method.

O.GLOBAL_ARRAYS_INTEG

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

O.NATIVE

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

O.OPERATE

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

O.REALLOCATION

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

O.RESOURCES

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

6.1.3 SERVICES

O.ALARM

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

O.CIPHER

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

O.KEY-MNGT

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

O.PIN-MNGT

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

Application note:

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

O.TRANSACTION

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

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

6.1.4 OBJECT DELETION

O.OBJ-DELETION

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

6.1.5 APPLET MANAGEMENT

O.DELETION

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

O.LOAD

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

Besides, for code loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the verification authority. This verification by the TOE shall occur during the loading or later during the install process.

Application note:

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

O.INSTALL

The TOE shall ensure that the installation of an applet performs as expected (See #.INSTALL for details).

Besides, for code loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the verification authority. If not performed during the loading process, this verification by the TOE shall occur during the install process.

O.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. Typically the card manager shall be in charge of the life cycle of the whole card, as well as that of the installed applications (applets). The card manager prevents 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.

6.1.6 SMART CARD PLATFORM

O.SCP.IC

The SCP shall provide all IC security features against physical attacks.

This security objective for the environment refers to the point (7) of the security aspect #.SCP:

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

O.SCP.RECOVERY

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

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

O.SCP.SUPPORT

The SCP shall support the TSFs of the TOE.

This security objective refers to the security aspects 2, 3, 4 and 5 of #.SCP:

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

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

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

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

6.2 Security objectives for the operational environment

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

OE.APPLET

No applet loaded post-issuance shall contain native methods.

(OE.SCP.IC, OE.SCP.RECOVERY, OE.CARD-MANAGEMENT and OE.SCP.SUPPORT have been turned into objectives for the TOE)

OE.VERIFICATION

All the bytecodes shall be verified at least once, before the loading, before the installation or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. See #.VERIFICATION for details.

Additionally, the applet shall follow all the recommendations, if any, mandated in the platform guidance for maintaining the isolation property of the platform.

Application Note:

Constraints to maintain the isolation property of the platform are provided by the platform developer in application development guidance. The constraints apply to all application code loaded in the platform.

OE.CODE-EVIDENCE

For application code loaded pre-issuance, evaluated technical measures implemented by the TOE or audited organizational measures must ensure that loaded application has not been changed since the code verifications required in OE.VERIFICATION.

For application code loaded post-issuance and verified off-card according to the requirements of OE.VERIFICATION, the verification authority shall provide digital evidence to the TOE that the application code has not been modified after the code verification and that he is the actor who performed code verification.

For application code loaded post-issuance and partially or entirely verified on-card, technical measures must ensure that the verification required in OE.VERIFICATION are performed. On-card bytecode verifier is out of the scope of this Security Target.

Application Note:

For application code loaded post-issuance and verified off-card, the integrity and authenticity evidence is achieved by electronic signature of the application code, after code verification, by the actor who performed verification.

6.3 Security objectives rationale

<u>Summary of changes in the rationale in comparison to the PP:</u>

<u>The newly introduced T.SECURE_DELETION (see 5) is covered by the security objective O.DELETION.</u>

<u>All occurrences of OE.CARD-MANAGEMENT in the rationale have been changed to O.CARD-MANAGEMENT.</u>

<u>All occurrences of OE.SCP.IC, OE.SCP.RECOVERY and OE.SCP.SUPPORT have been changed to O.SCP.IC, O.SCP.RECOVERY and O.SCP.SUPPORT, respectively.</u>

The assumption A.DELETION has been removed as stated in 5, so it doesn't need to be covered by OE.CARD-MANAGEMENT any more.

The threat T.EXE-CODE-REMOTE and the objective O.REMOTE have been removed, so they do not appear in the rationale anymore.

6.3.1 Threats

6.3.1.1 Confidentiality

T.CONFID-APPLI-DATA

This threat is countered by the security objective for the operational environment regarding bytecode verification (OE.VERIFICATION). It is also covered by the isolation commitments stated in the (O.FIREWALL) objective. It relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

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

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

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

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

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

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

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

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

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

T.CONFID-JCS-DATA This threat is covered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) security objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID).

Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

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

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

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

6.3.1.2 INTEGRITY

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

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

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

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

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

T.INTEG-APPLI-DATA This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective.

This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.

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

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

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

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

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

that buffer is ensured by the objective O.GLOBAL_ARRAYS_INTEG.

Finally, any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the O.REALLOCATION objective.

That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused.

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

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

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

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

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

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

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

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

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

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

6.3.1.3 IDENTITY USURPATION

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

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

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

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

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

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

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

6.3.1.4 UNAUTHORIZED EXECUTION

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

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

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

6.3.1.5 DENIAL OF SERVICE

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

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

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

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

6.3.1.6 CARD MANAGEMENT

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

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

T.SECURE_DELETION This threat is covered by the O.DELETION objective which ensures that deletion through the card manager is secure.

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

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

6.3.1.7 SERVICES

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

6.3.1.8 MISCELLANEOUS

T.PHYSICAL Covered by O.SCP.IC. Physical protections rely on the underlying platform which is defined to be part of the TOE.

6.3.2 ORGANISATIONAL SECURITY POLICIES

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

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

6.3.3 ASSUMPTIONS

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

(A.DELETION has been removed)

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

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

6.3.4 SPD and security objectives

Threats	Security Objectives	Rationale
T.CONFID-APPLI-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.REALLOCATION	Section 6.3.1
T.CONFID-JCS-CODE	OE.VERIFICATION, O.CARD-MANAGEMENT, O.NATIVE	Section 6.3.1
T.CONFID-JCS-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM	Section 6.3.1
T.INTEG-APPLI-CODE	O.CARD-MANAGEMENT, OE.VERIFICATION, O.NATIVE, OE.CODE-EVIDENCE	Section 6.3.1
T.INTEG-APPLI-CODE.LOAD	O.LOAD, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	Section 6.3.1
T.INTEG-APPLI-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.GLOBAL_ARRAYS_INTEG, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.REALLOCATION, OE.CODE-EVIDENCE	Section 6.3.1
T.INTEG-APPLI-DATA.LOAD	O.LOAD, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	Section 6.3.1
T.INTEG-JCS-CODE	O.CARD-MANAGEMENT, OE.VERIFICATION, O.NATIVE, OE.CODE-EVIDENCE	Section 6.3.1

T.INTEG-JCS-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM, OE.CODE-EVIDENCE	Section 6.3.1
T.SID.1	O.CARD-MANAGEMENT, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.INSTALL, O.SID	Section 6.3.1
T.SID.2	O.SCP.RECOVERY, O.SCP.SUPPORT, O.SID, O.OPERATE, O.FIREWALL, O.INSTALL	Section 6.3.1
T.EXE-CODE.1	OE.VERIFICATION, O.FIREWALL	Section 6.3.1
T.EXE-CODE.2	OE.VERIFICATION	Section 6.3.1
T.NATIVE	OE.VERIFICATION, OE.APPLET, O.NATIVE	Section 6.3.1
T.RESOURCES	O.INSTALL, O.OPERATE, O.RESOURCES, O.SCP.RECOVERY, O.SCP.SUPPORT	Section 6.3.1
T.DELETION	O.DELETION, O.CARD-MANAGEMENT	Section 6.3.1
T.SECURE_DELETION	O.DELETION	
T.INSTALL	O.INSTALL, O.LOAD, O.CARD-MANAGEMENT	Section 6.3.1
T.OBJDELETION	O.OBJ-DELETION	Section 6.3.1
T.PHYSICAL	O.SCP.IC	Section 6.3.1

Table 1 Threats and and objectives - Coverage

Security Objectives	Threats
O.SID	T.CONFID-APPLI-DATA, T.CONFID-JCSDATA, T.INTEG-APPLI-DATA, T.INTEGJCS-DATA, T.SID.1, T.SID.2
O.FIREWALL	T.CONFID-APPLI-DATA, T.CONFID-JCSDATA, T.INTEG-APPLI-DATA, T.INTEGJCS-DATA, T.SID.1, T.SID.2, T.EXECODE.1
O.GLOBAL_ARRAYS_CONFID	T.CONFID-APPLI-DATA, T.SID.1
O.GLOBAL_ARRAYS_INTEG	T.INTEG-APPLI-DATA, T.SID.1
O.NATIVE	T.CONFID-JCS-CODE, T.INTEG-APPLICODE, T.INTEG-JCS-CODE, T.NATIVE

O.OPERATE	T.CONFID-APPLI-DATA, T.CONFID-JCSDATA, T.INTEG-APPLI-DATA, T.INTEGJCS-DATA, T.SID.2, T.RESOURCES
O.REALLOCATION	T.CONFID-APPLI-DATA, T.INTEG-APPLIDATA
O.RESOURCES	T.RESOURCES
O.ALARM	T.CONFID-APPLI-DATA, T.CONFID-JCSDATA, T.INTEG-APPLI-DATA, T.INTEGJCS-DATA
O.CIPHER	T.CONFID-APPLI-DATA, T.INTEG-APPLIDATA
O.KEY-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLIDATA
O.PIN-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLIDATA
O.TRANSACTION	T.CONFID-APPLI-DATA, T.INTEG-APPLIDATA
O.OBJ-DELETION	T.OBJ-DELETION
O.DELETION	T.DELETION, T.SECURE_DELETION
O.LOAD	T.INTEG-APPLI-CODE.LOAD, T.INTEGAPPLI-DATA.LOAD, T.INSTALL
O.INSTALL	T.SID.1, T.SID.2, T.RESOURCES, T.INSTALL
OE.APPLET	T.NATIVE
O.CARD-MANAGEMENT	T.CONFID-APPLI-DATA, T.CONFID-JCSCODE, T.CONFID-JCS-DATA, T.INTEGAPPLI-CODE, T.INTEG-APPLICODE.LOAD, T.INTEG-APPLI-DATA, T.INTEG-APPLI-DATA.LOAD, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.SID.1, T.DELETION, T.INSTALL
O.SCP.IC	T.PHYSICAL
O.SCP.RECOVERY	T.CONFID-APPLI-DATA, T.CONFID-

	JCSDATA, T.INTEG-APPLI-DATA, T.INTEGJCS-DATA, T.SID.2, T.RESOURCES
O.SCP.SUPPORT	T.CONFID-APPLI-DATA, T.CONFID-JCSDATA, T.INTEG-APPLI-DATA, T.INTEGJCS-DATA, T.SID.2, T.RESOURCES
OE.VERIFICATION	T.CONFID-APPLI-DATA, T.CONFID-JCSCODE, T.CONFID-JCS-DATA, T.INTEGAPPLI-CODE, T.INTEG-APPLI-DATA, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.EXE-CODE.1, T.EXE-CODE.2, T.NATIVE
OE.CODE-EVIDENCE	T.INTEG-APPLI-CODE, T.INTEG-APPLICODE.LOAD, T.INTEG-APPLI-DATA, T.INTEG-APPLI-DATA.LOAD, T.INTEG-JCSCODE, T.INTEG-JCS-DATA

Table 2: Security Objectives and Threats – Coverage

Organisational Security Policies	Security Objectives	Rationale
OSP.VERIFICATION	OE.VERIFICATION, O.LOAD, OE.CODE-EVIDENCE	Section 6.3.2

Table 3: OSPs and Security Objectives – Coverage

Security Objectives	Organisational Security Policies
O.SID	
O.FIREWALL	
O.GLOBAL_ARRAYS_CONFID	
O.GLOBAL_ARRAYS_INTEG	
O.NATIVE	
O.OPERATE	
O.REALLOCATION	
O.RESOURCES	
O.ALARM	

O.CIPHER	
O.KEY-MNGT	
O.PIN-MNGT	
O.TRANSACTION	
O.OBJ-DELETION	
O.DELETION	
O.LOAD	OSP.VERIFICATION
O.INSTALL	
OE.APPLET	
O.CARD-MANAGEMENT	
O.SCP.IC	
O.SCP.RECOVERY	
O.SCP.SUPPORT	
OE.VERIFICATION	OSP.VERIFICATION
OE.CODE-EVIDENCE	OSP.VERIFICATION

Table 4: Security Objectives and OSPs - Coverage

Please note that the assumption A.DELETION has been removed (changed to the threat T.SECURE_DELETION) and does therefore not appear in this table.

Assumptions	Security Objectives for the Operational Environment	Rationale
A.APPLET	OE.APPLET	Section 6.3.3
A.VERIFICATION	OE.VERIFICATION, OE.CODE-EVIDENCE	Section 6.3.3

Table 5: Assumptions and Security Objectives for the Operational Environment –Coverage

Please note that the assumption A.DELETION has been removed and doesn’t need to be covered anymore. The objectives OE.SCP.IC/RECOVERY/SUPPORT have been changed to objectives on the TOE and do not appear in this table.

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

Table 6: Security Objectives for the Operational Environment and Assumptions – Coverage

7 Extended Components Definition

This security target uses a component defined as extensions to CC part 2 as defined in the protection profile [PP0084].

7.1 Definition of the Family FCS_RNG

To define the IT security functional requirements of the TOE an additional family (FCS_RNG) of the Class FCS (cryptographic support) is defined here. This family describes the functional requirements for random number generation used for cryptographic purposes.

FCS_RNG Generation of random numbers

Family behaviour

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

Component levelling:

FCS_RNG: Generation of random numbers — 1

FCS_RNG.1

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

Management: FCS_RNG.1

There are no management activities foreseen.

Audit: FCS_RNG.1

There are no actions defined to be auditable.

FCS_RNG.1 Random number generation

Hierarchical to: No other components

Dependencies: No dependencies

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

FCS_RNG.1.2 The TSF shall provide [selection: *bits, octets of bits, numbers [assignment: format of the numbers]*] that meet [assignment: *a defined quality metric*].

8 Security Functional Requirements

8.1 Security functional requirements

This section states the security functional requirements for the TOE, organised in groups. In addition to the groups defined in [JCSPP], 2 groups have been added (CMGR and SCP).

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

and low-level cryptographic processing.

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

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

Subject/Object/Information	Description
S.ADEL	The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([JCRE22], §11), but its role asks anyway for a specific treatment from the security viewpoint. This subject is unique and is involved in the ADEL security policy defined in 8.1.3.
S.APPLET	Any applet instance.
S.BCV	The bytecode verifier (BCV), which acts on behalf of the verification authority who is in charge of the bytecode verification of the packages. This subject is involved in the PACKAGE LOADING security policy defined in 8.1.6.
S.CAD	The CAD[2] represents off-card entity that communicates with the S.INSTALLER. If the TOE provides JCRMI functionality, this subject can also plays the role of the actor that requests, by issuing commands to the card, for RMI[3] services.
S.CARDMANAGER	The Card Manager charges Installer and Applet Deletion Manager to perform card content management operations (content loading, installation and deletion).
S.INSTALLER	The installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the loading of packages and installation of applets.
S.JCRE	The runtime environment under which Java programs in a smart card are executed.
S.JCVM	The bytecode interpreter that enforces the firewall at runtime.
S.LOCAL	Operand stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.
S.MEMBER	Any object's field, static field or array position.
S.PACKAGE	A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of Java Card technology, it defines either a user library, or one or several applets.

2 The acronym CAD is used here and throughout this security target to refer to both types of card readers – the conventional Card Acceptance Device (CAD) for contacted I/O interfaces and the Proximity Coupling Device (PCD) for contactless interfaces.

3 Application note by the ST author: RMI is not supported by the TOE so that RMI services will not be requested by S.CAD.

S.SPY	Any subject that potentially observe security critical operations to disclose keys and PINs.

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

O.APPLET	Any installed applet, its code and data.
O.CODE_PKG	The code of a package, including all linking information. On the Java Card platform, a package is the installation unit.
O.JAVAOBJECT	Java class instance or array. It should be noticed that KEYS, PIN, arrays and applet instances are specific objects in the Java programming language.

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

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

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

Security attribute	**Description / Value**
Active Applets	The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels.
Applet Selection Status	"Selected" or "Deselected".
Applet's version number	The version number of an applet (package) indicated in the export file.
Context	Package AID or "Java Card RE".
Currently Active Context	Package AID or "Java Card RE".
Dependent package AID	Allows the retrieval of the Package AID and Applet's version number ([JCVM22], §4.5.2).
LC Selection Status	Multiselectable, Non-multiselectable or "None".
LifeTime	CLEAR_ON_DESELECT or PERSISTENT (*).
Owner	The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the package).
Package AID	The AID of each package indicated in the export file.

Registered Applets	The set of AID of the applet instances registered on the card.
Resident Packages	The set of AIDs of the packages already loaded on the card.
Selected Applet Context	Package AID or "None".
Sharing	Standards, SIO, Java Card RE entry point or global array.
Static References	Static fields of a package may contain references to objects. The Static References attribute records those references.

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

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

Operation	**Description**
OP.ARRAY_ACCESS(O.JAVAOBJECT, field)	Read/Write an array component.
OP.CREATE(Sharing, LifeTime) (*)	Creation of an object (new or makeTransient call).
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed applet and its objects, either logically or physically.
OP.DELETE_PCKG(O.CODE_PKG,...)	Delete a package, either logically or physically.
OP.DELETE_PCKG_APPLET(O.CODE_PKG,...)	Delete a package and its installed applets, either logically or physically.
OP.INSTANCE_FIELD(O.JAVAOBJECT, field)	Read/Write a field of an instance of a class in the Java programming language.
OP.INVK_VIRTUAL(O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object).
OP.INVK_INTERFACE(O.JAVAOBJECT, method, arg1,...)	Invoke an interface method.
OP.JAVA(...)	Any access in the sense of [JCRE22], §6.2.8. It stands for one

	of the operations OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW, OP.TYPE_ACCESS.
OP.PUT(S1,S2,I)	Transfer a piece of information I from S1 to S2.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see [JCRE22], §6.2.8.7).
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	Invoke checkcast or instanceof on an object in order to access to classes (standard or shareable interfaces objects).

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

8.1.1 COREG_LC SECURITY FUNCTIONAL REQUIREMENTS

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

8.1.1.1 Firewall Policy

FDP_ACC.2/FIREWALL Complete access control

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

Refinement:

The operations involved in the policy are:

o OP.CREATE,

o OP.INVK_INTERFACE,

o OP.INVK_VIRTUAL,

o OP.JAVA,

o OP.THROW,

o OP.TYPE_ACCESS.

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

Application note:

Accessing array's components of a static array, and more generally fields and methods of static objects, is an access to the corresponding O.JAVAOBJECT.

FDP_ACF.1/FIREWALL Security attribute based access control

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

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

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

o **R.JAVA.1 ([JCRE22], §6.2.8): S.PACKAGE may freely perform OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW or OP.TYPE_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".**

o **R.JAVA.2 ([JCRE22], §6.2.8): S.PACKAGE may freely perform OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE or OP.THROW upon any O.JAVAOBJECT whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if O.JAVAOBJECT's Context attribute has the same value as the active context.**

o **R.JAVA.3 ([JCRE22], §6.2.8.10): S.PACKAGE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT whose Sharing attribute has value "SIO" only if O.JAVAOBJECT is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.**

o **R.JAVA.4 ([JCRE22], §6.2.8.6): S.PACKAGE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute**

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

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

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

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

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

o **1) The subject S.JCRE can freely perform OP.JAVA(") and OP.CREATE, with the exception given in FDP_ACF.1.4/FIREWALL, provided it is the Currently Active Context.**

o **2) The only means that the subject S.JCVM shall provide for an application to execute native code is the invocation of a Java Card API method (through OP.INVK_INTERFACE or OP.INVK_VIRTUAL)**.

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

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

Application note:

FDP_ACF.1.4/FIREWALL:

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

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

The Sharing attribute defines four categories of objects:

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

When a new object is created, it is associated with the Currently Active Context. But the object is owned by the applet instance within the Currently Active Context when the object is instantiated ([JCRE22], §6.1.3). An object is owned by an applet instance, by the JCRE or by the package library where it has been defined (these latter objects are arrays that initialize static fields of packages).

([JCRE22], Glossary) Selected Applet Context; The Java Card RE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command with this applet's AID, the Java Card RE makes this applet the Selected Applet Context. The Java Card RE sends all APDU commands to the Selected Applet Context.

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

([JCRE22], §6.1.2.1) At any point in time, there is only one active context within the Java Card VM (this is called the Currently Active Context).

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

The Java Card platform, version 2.2.x and version 3 Classic Edition, introduces the possibility for an applet instance to be selected on multiple logical channels at the same time, or accepting other applets belonging to the same package being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same package are either all multiselectable or not ([JCVM22], §2.2.5). Therefore, the selection mode is regarded as an attribute of packages. No selection mode is defined for a library package.

An applet instance will be considered an active applet instance if it is currently selected in at least one logical channel. An applet instance is the currently selected applet instance only if it is processing the current command. There is only one currently selected applet instance at a given time. ([JCRE22], §4).

FDP_IFC.1/JCVM Subset information flow control

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

Application note:

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

FDP_IFF.1/JCVM Simple security attributes

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

Subjects	Security attributes
S.JCVM	**Currently Active Context**

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

o **An operation OP.PUT(S1, S.MEMBER, I.DATA) is allowed if and only if the Currently Active Context is "Java Card RE";**

o **other OP.PUT operations are allowed regardless of the Currently Active Context's value**.

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

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

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

Application note:

The storage of temporary Java Card RE-owned objects references is runtime-enforced ([JCRE22], §6.2.8.1-3).

This policy essentially applies to the execution of bytecode. Native methods, the Java Card RE itself and possibly some API methods are granted specific rights or limitations through the FDP_IFF.1.3/JCVM to FDP_IFF.1.5/JCVM elements.

FDP_RIP.1/OBJECTS Subset residual information protection

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

Application note:

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

FMT_MSA.1/JCRE Management of security attributes

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

Application note:

The modification of the Selected Applet Context is performed in accordance with the rules given in [JCRE22], §4 and [JCVM22], §3.4.

FMT_MSA.1/JCVM Management of security attributes

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

Application note:

The modification of the Currently Active Context is performed in accordance with the rules given in [JCRE22], §4 and [JCVM22], §3.4.

FMT_MSA.2/FIREWALL_JCVM Secure security attributes

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

Application note:

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

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

FMT_MSA.3/FIREWALL Static attribute initialisation

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

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

Application note:

FMT_MSA.3.1/FIREWALL

- Objects' security attributes of the access control policy are created and initialized at the creation of the object or the subject. Afterwards, these attributes are no longer mutable (FMT_MSA.1/JCRE). At the creation of an object (OP.CREATE), the newly created object, assuming that the FIREWALL access control SFP permits the operation, gets its Lifetime and Sharing attributes from the parameters of the operation; on the contrary, its Context attribute has a default value, which is its creator's Context attribute and AID respectively ([JCRE22], §6.1.3). There is one default value for the Selected Applet Context that is the default applet identifier's Context, and one default value for the Currently Active Context that is "Java Card RE".
- The knowledge of which reference corresponds to a temporary entry point object or a global array and which does not is solely available to the Java Card RE (and the Java Card virtual machine).

FMT_MSA.3.2/FIREWALL

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

FMT_MSA.3/JCVM Static attribute initialisation

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

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

FMT_SMF.1 Specification of Management Functions

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

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

FMT_SMR.1 Security roles

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

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

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

8.1.1.2 APPLICATION PROGRAMMING INTERFACE

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

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

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

FCS_CKM.1 Cryptographic key generation

FCS_CKM.1.1/RSA The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm: **G&D RSA-CRT and RSA Key generator**[4] and specified cryptographic key sizes **512 up to 4096 bit (RSA-CRT) and 512 up to 2048 bit (RSA)** that meet the following: **list of standards: [FIPS 186-3] Section 5.1 and B.3.3.**

FCS_CKM.1.1/ECC The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm: **G&D EC Key Generator**[5] and specified cryptographic key sizes **corresponding to the used elliptic curves secp {160, 192, 224, 256, 320, 384, 521}r1, brainpoolP{160, 192, 224, 256, 320, 384, 512}r1 and brainpoolP{160, 192, 224, 256, 320, 384, 512}t1** that meet the following: **list of standards: secp curves according [FIPS 186-3] chapter B.4.1 and D.1.2. and brainpool curves according to [RFC5639] chapter 3.**

---

[4] The TOE generates keys for RSA-CRT and RSA. For these algorithms, the key generator used by the TOE is the random number generator that meets [AIS20], Section 4.9 Class DRG.4 (see: FCS_RNG.1).

[5] The G&D EC Key generator generates keys for the Diffie-Hellman key derivation compliant to PKCS#3, based on an ECDH protocol compliant to ISO 15946 and for ECDSA for example. Therefore the API GDKeyagreement with ECDH is part of this SFR.

FCS_CKM.1.1/3DES The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm: **G&D 3DES Key Generator** and specified cryptographic key sizes **112, 168 bits** that meet the following: **list of standards: [SP800-67] Sections 3.4.1 and 3.4.2,**

FCS_CKM.1.1/AES The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm: **G&D AES Key generator** and specified cryptographic key sizes **128, 192 and 256 bits** that meet the following: **list of standards: [FIPS 197], chapter 3.1 and 5.**

Application note:

- The keys are FIPS 197 generated and diversified in accordance with [JCAPI304] specification in classes KeyBuilder and KeyPair (at least Session key generation).
- This component is instantiated according to the version of the Java Card API applying to the security target and the implemented algorithms ([JCAPI304]).

FCS_CKM.2 Cryptographic key distribution

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **all set-methods of RSAPrivateCrtKey, RSAPrivateKey, RSAPublicKey, AESKey, DESKey, ECPrivateKey, ECPublicKey** that meets the following **list of standards: [JCAPI304], class javacard.security.KeyBuilder.**

Application note:

- Command SetKEY meets [JCAPI304] specification.
- This component is instantiated according to the version of the Java Card API applying to this security target and the implemented algorithms ([JCAPI304]).

FCS_CKM.3 Cryptographic key access

FCS_CKM.3.1 The TSF shall perform **key access to the 3-DES/RSA/AES/ECC keys** in accordance with a specified cryptographic key access method: **all get-methods of class RSAPrivateCrtKey, RSAPrivateKey, RSAPublicKey, AESKey, DESKey, ECPrivateKey, ECPublicKey** that meets the following: **list of standards: [JCAPI304], class javacard.security.KeyBuilder.**

Application note:

- The keys are accessed as specified in **[JCAPI304]** Key class.
- This component is instantiated according to the version of the Java Card API applicable to this security target and the implemented algorithms ([JCAPI304]).

FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **Key.clearKey() and overwriting the keys with zeros** that meets the following: **list of standards: [JCAPI304], class javacard.security.KeyBuilder.**

Application note:

- The keys are reset as specified in [JCAPI304] Key class, with the method clearKey(). Any access to a cleared key for ciphering or signing throws an exception.
- This component is instantiated according to the version of the Java Card API applicable to this security target and the implemented algorithms ([JCAPI304]).

FCS_COP.1 Cryptographic operation

FCS_COP.1.1/RSA-CRT-SIGN

The TSF shall perform **signature generation** in accordance with a specified cryptographic algorithm **RSA-CRT** and cryptographic key sizes **512 up to 4096 bit** that meet the following**: scheme 1 of [ISO9796-2] chapter 8, [RSA] (RSASSA-PKCS1-v15) chapter 8, [RSASSA-PSS] and [RSA-SHA-RFC2409].**

FCS_COP.1.1/RSA-SIGN

The TSF shall perform **signature generation** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **512 up to 2048 bit** that meet the following**: scheme 1 of [ISO9796-2] chapter 8.2 [RSA] (RSASSA-PKCS1-v15) chapter 8, [RSASSA-PSS] and [RSA-SHA-RFC2409]**

FCS_COP.1.1/RSA-VERI

The TSF shall perform **signature verification** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **512 up to 2048 bit** that meet the following**: [ISO9796-2] chapter 8.2 [RSA] (RSASSA-PKCS1-v15) chapter 8, [RSASSA-PSS] and [RSA-SHA-RFC2409]**

FCS_COP.1.1/MAC-DES

The TSF shall perform **MAC generation and verification** in accordance with a specified cryptographic algorithm **DES CBC-MAC and DES Retail-MAC** and cryptographic key sizes **112, 168 bit** that meet the following: **[ISO9797-1] (CBC-MAC, Retail MAC) chapter 7.2 and chapter 7.4.**

FCS_COP.1.1/MAC-AES

The TSF shall perform **MAC generation and verification** in accordance with a specified cryptographic algorithm **AES CBC-MAC** and cryptographic key sizes **128, 192, 256 bit** that meet the following: **[ISO9797-1] (CBC-MAC) chapter 7.2.**

FCS_COP.1.1/CMAC-AES

The TSF shall perform **MAC generation and verification** in accordance with a specified cryptographic algorithm **AES CMAC** and cryptographic key sizes **128, 192, 256 bit** that meet the following: **[SP800-38b] (CMAC) chapter 6.**

FCS_COP.1.1/3DES

The TSF shall perform **encryption/decryption** in accordance with a specified cryptographic algorithm **3-DES in CBC/ ECB mode** and cryptographic key sizes **112, 168 bit** that meet the following **list of standards:[FIPS46-3] (3DES) chapter "TRIPLE DATA ENCRYPTION ALGORITHM" for 3-DES, [SP800-38a] chapter 6.2 for the CBC mode, and [SP800-38a] (ECB) chapter 6.1 for the ECB mode and [ISO9797-1] padding method M1 and M2 and [PKCS5] padding for the CBC/ECB mode.**

FCS_COP.1.1/AES

The TSF shall perform **encryption/decryption** in accordance with a specified cryptographic algorithm **AES in CBC/ECB mode** and cryptographic key sizes **128, 192, 256 bit** that meet the following **list of standards: [FIPS 197] (AES) chapter 5 for AES, [SP800-38a] (CBC) chapter 6.2 for the CBC mode, and [SP800-38a] (ECB) chapter 6.1 for the ECB mode and [ISO9797-1] padding method M1 and M2 and [PKCS5] padding for the CBC/ECB mode.**

FCS_COP.1.1/RSA-DEC

The TSF shall perform **decryption** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **512 up to 2048 bit** that meet the following **list of standards: RFC 3447 [RSA] chapter 5.1 and chapter 7.2 and without encoding (RSASP1** [PKCS1]**[PKCS1] chapter 5.2, RSAVP1 [RFC2437] chapter 5.2.2)..**

FCS_COP.1.1/RSA-CRT-DEC

The TSF shall perform **decryption** in accordance with a specified cryptographic algorithm **RSA-CRT** and cryptographic key sizes **512 up to 4096 bit** that meet the following **list of standards: RFC 3447 [RSA] chapter 5.1 and chapter 7.2 and without encoding (RSASP1** [PKCS1]**[PKCS1] chapter 5.2, RSAVP1 [RFC2437] chapter 5.2.2).**

FCS_COP.1.1/RSA-ENC

The TSF shall perform **encryption** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **512 up to 2048 bit** that meet the following **list of standards: RFC 3447[RSA] chapter 5.1 and chapter 7.2 (RSAAES-PKCS1-v1_5) and without encoding (RSASP1** [PKCS1]**[PKCS1] chapter 5.2, RSAVP1 [RFC2437] chapter 5.2.2).**

FCS_COP.1.1/ECDSA-SIGN

The TSF shall perform **signature generation** in accordance with a specified cryptographic algorithm **ECDSA-FP** and cryptographic key sizes **corresponding to the used elliptic curves secp{160, 192, 224, 256, 320, 384, 521}r1[RSA-SHA-RFC2409]** RFC 2409, D. Harkins, D. Carrel, “The Internet Key Exchange (IKE), November 1998

[SEC2]**, brainpoolP{160, 192, 224, 256, 320, 384, 512}r1 and brainpoolP{160, 192, 224, 256, 320, 384, 512}t1 [RFC5639]** that meet the following **standard: [TR-3111], (ECDSA), chapter 4.2.1.**

FCS_COP.1.1/ECDSA-VERI

The TSF shall perform **signature verification** in accordance with a specified cryptographic algorithm **ECDSA-FP** and cryptographic key sizes **corresponding to the used elliptic curves secp{160, 192, 224, 256, 320, 384, 521}r1[RSA-SHA-RFC2409]** RFC 2409, D. Harkins, D. Carrel, “The Internet Key Exchange (IKE), November 1998

[SEC2]**, brainpoolP{160, 192, 224, 256, 320, 384, 512}r1 and brainpoolP{160, 192, 224, 256, 320, 384, 512}t1[RFC5639]** that meet the following **standard: [TR-3111], (ECDSA), chapter 4.2.1.**

FCS_COP.1.1/HASH

The TSF shall perform **hash calculation** in accordance with a specified cryptographic algorithm **SHA-1, SHA-224, SHA-256, SHA-384, SHA-512** and cryptographic key sizes **none** that meet the following **list of standards: chapter 6.1 - 6.5 [FIPS180-4] (SHA).**

Random Number Generation (FCS_RNG.1)

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

FCS_RNG.1 Quality metric for random numbers

Hierarchical to: No other components.

Dependencies: No dependencies.

Refinement:

FCS_RNG.1.1 The TSF shall provide a **hybrid deterministic**[6] random number generator that implements[7]:

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

Refinement:

FCS_RNG.1.2 The TSF shall provide random numbers[8] that meet[9]:

- (**DRG.4.6) The RNG generates output for which two strings of bit length 128 are mutually different with probability 1 - 2^-128.**

---

6 [selection: physical, non-physical true,deterministic, hybrid physical, hybrid deterministic]

7 [assignment: list of security capabilities]

8 [selection: *bits, octets of bits, numbers [assignment: format of the numbers]*]

9 [assignment: a defined quality metric]

- **<u>(DRG.4.7) Statistical test suites cannot practically distinguish the random number from output sequences of an ideal RNG. The random numbers pass test procedure A as defined in AIS20/31.</u>**

FDP_RIP.1/ABORT Subset residual information protection

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

Application note:

The events that provoke the de-allocation of a transient object are described in [JCRE304].

FDP_RIP.1/APDU Subset residual information protection

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

Application note:

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

FDP_RIP.1/bArray Subset residual information protection

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

Application note:

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

FDP_RIP.1/KEYS Subset residual information protection

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

Application note:

- The javacard.security & javacardx.crypto packages do provide secure interfaces to the cryptographic buffer in a transparent way. See javacard.security.KeyBuilder and Key interface of [JCAPI304].

FDP_RIP.1/TRANSIENT Subset residual information protection

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

Application note:

- The events that provoke the de-allocation of any transient object are described in [JCRE22], §5.1.
- The clearing of CLEAR_ON_DESELECT objects is not necessarily performed when the owner of the objects is deselected. In the presence of multiselectable applet instances, CLEAR_ON_DESELECT memory segments may be attached to applets that are active in different logical channels. Multiselectable applet instances within a same package must share the transient memory segment if they are concurrently active ([JCRE304]).

FDP_ROL.1/FIREWALL Basic rollback

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

FDP_ROL.1.2/FIREWALL The TSF shall permit operations to be rolled back within the **scope of a select(), deselect(), process(), install() or uninstall() call, notwithstanding the restrictions given in [JCRE304], within the bounds of the Commit Capacity ([JCRE304]), and those described in [JCAPI304]**.

Application note:

Transactions are a service offered by the APIs to applets. It is also used by some APIs to guarantee the atomicity of some operation. This mechanism is implemented in Java Card platform and uses the transaction mechanism offered by the underlying platform for atomic operations. Some operations of the API are not conditionally updated, as documented in **[JCAPI304]** (see for instance, PIN-blocking, PIN-checking, update of Transient objects).

8.1.1.3 CARD SECURITY MANAGEMENT

FAU_ARP.1 Security alarms

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

- **throw an exception,**
- **lock the card session,**
- **reinitialize the Java Card System and its data,**
- **other actions: Card Lock / Application Lock**

upon detection of a potential security violation.

Refinement:

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

- CAP file inconsistency,
- typing error in the operands of a bytecode,
- applet life cycle inconsistency,
- card tearing (unexpected removal of the Card out of the CAD) and power failure,
- abort of a transaction in an unexpected context, (see abortTransaction(),[JCAPI304] and ([JCRE304])
- violation of the Firewall or JCVM SFPs,
- unavailability of resources,
- array overflow,

- **improper execution of sub-functions monitored by flow control.**

Application note:

- The "locking of the card session" may not appear in the policy of the card manager. Such measure is only 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 impossible.
- The locking is implemented at the level of the Java Card System as a denial of service (through some systematic "fatal error" message or return value) that lasts up to the next "RESET" event, without affecting other components of the card (such as the card manager). Finally, because the installation of applets is a sensitive process, security alerts in this case are also be carefully considered herein.

FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for **integrity errors** on all objects, based on the following attributes: **checksum integrity of cryptographic keys, PIN values and their associated security attributes.**

.

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **bring the card into a secure state.**

Application note:

- Although no such requirement is mandatory in the Java Card specification, at least an exception is raised upon integrity errors detection on cryptographic keys, PIN values and their associated security attributes. **Cryptographic keys and PIN objects are considered as described in FDP_SDI.2.1**.
- **Integrity errors in the code of the Java Card applets are monitored**.
- For integrity sensitive application, their data is 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.
- A dedicated library **is** implemented and made available to developers to achieve better security for specific objects, following the same pattern that already exists in cryptographic APIs.

FPR_UNO.1 Unobservability

FPR_UNO.1.1 The TSF shall ensure that **S.SPY** are unable to observe the operation **cryptographic operations / comparison operations** on **key values / PIN values** by **S.JCRE, S.Applet**.

FPT_FLS.1 Failure with preservation of secure state

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

Application note:

The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE22], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE22]). Behaviour of the TOE on power loss and reset is described in [JCRE22], §3.6 and §7.1. Behaviour of the TOE on RF signal loss is described in [JCRE22], §3.6.1.

FPT_TDC.1 Inter-TSF basic TSF data consistency

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

FPT_TDC.1.2 The TSF shall use

- **the rules defined in [JCVM22] specification,**
- **the API tokens defined in the export files of reference implementation,**
- **the ISO 7816-6 rules**

when interpreting the TSF data from another trusted IT product.

Application note:

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

FPT_TST.1 TSF testing

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.

Application note: TSF-testing is not mandatory in [JCRE22], but appears in most of security requirements documents for masked applications.

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

FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of **stored TSF executable code.**

8.1.1.4 AID MANAGEMENT

FIA_ATD.1/AID User attribute definition

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

- **Package AID,**
- **Applet's version number,**
- **Registered applet AID,**
- **Applet Selection Status (**[JCVM22]**, §6.5)**.

Refinement:

"Individual users" stand for applets.

FIA_UID.2/AID User identification before any action

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

Application note:

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

FIA_USB.1/AID User-subject binding

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

FIA_USB.1.2/AID The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **rules defined in FMT_MSA.2/FIREWALL_JCVM and FMT_MSA.3.1/FIREWALL.**

FIA_USB.1.3/AID The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: **rules defined in FMT_MSA.3.1/FIREWALL.**

Application note:

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

FMT_MTD.1/JCRE Management of TSF data

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

Application note:

- The installer and the Java Card RE manage other TSF data such as the applet life cycle or CAP files. Objects in the Java programming language may also try to query AIDs of installed applets through the lookupAID(...) API method.

- The installer, applet deletion manager or even the card manager is granted the right to modify the list of registered applets' AIDs (possibly needed for installation and deletion; see #.DELETION and #.INSTALL).

FMT_MTD.3/JCRE Secure TSF data

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

8.1.2 INSTG SECURITY FUNCTIONAL REQUIREMENTS

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

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

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

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

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

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

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

Package loading is allowed only if, for each dependent package, its AID attribute is equal to a resident package AID attribute, the major (minor) Version attribute associated to the dependent package is lesser than or equal to the major (minor) Version attribute associated to the resident package ([JCVM22]**, §4.5.2).**

Application note:

FDP_ITC.2.1/Installer:

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

FDP_ITC.2.3/Installer:

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

FDP_ITC.2.4/Installer:

- Each package contains a package Version attribute, which is a pair of major and minor version numbers ([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 retrieval of these identifications. 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.

FDP_ITC.2.5/Installer:

- A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs.

- The intent of this rule is to ensure the binary compatibility of the package with those already on the card ([JCVM22], §4.4).

FMT_SMR.1/Installer Security roles

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

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

FPT_FLS.1/Installer Failure with preservation of secure state

FPT_FLS.1.1/Installer The TSF shall preserve a secure state when the following types of failures occur: **the installer fails to load/install a package/applet as described in <u>[JCRE22] , §11.1.5.</u>**

Application note:

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

FPT_RCV.3/Installer Automated recovery without undue loss

FPT_RCV.3.1/Installer When automated recovery from **<u>power loss</u>** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2/Installer For **<u>reset, insufficient memory, failure in cryptographic safeguarding, package references (versions) mismatching</u>**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/Installer The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **<u>0%</u>** for loss of TSF data or objects under the control of the TSF.

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

Application note:

FPT_RCV.3.1/Installer:

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

FPT_RCV.3.2/Installer:

- Should the installer fail during loading/installation of a package/applet, it has to revert to a "consistent and secure state". The Java Card RE has some clean up duties as well; see [JCRE22], §11.1.5 for possible scenarios. This component includes among the listed failures the deletion of a package/applet. See ([JCRE22], 11.3.4) for possible scenarios.
- Other events such as the unexpected tearing of the card, power loss, and so on, are partially handled by the underlying hardware platform (see [PP0035]) and, from the TOE's side, by events "that clear transient objects" and transactional features. See FPT_FLS.1.1, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT and FDP_ROL.1/FIREWALL.

FPT_RCV.3.3/Installer:

- First, the SCP ensures the atomicity of updates for fields and objects, and a power-failure during a transaction or the normal runtime does not create the loss of otherwise permanent data, in the sense that memory on a smart card is essentially persistent with this respect (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 TSF scope is limited to the same restrictions of the transaction mechanism.

8.1.3 ADELG SECURITY FUNCTIONAL REQUIREMENTS

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

FDP_ACC.2/ADEL Complete access control

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

Refinement:

The operations involved in the policy are:

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

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

FDP_ACF.1/ADEL Security attribute based access control

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

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

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

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

- **the owner of O is a registered applet instance A (O is reachable from A),**

- **(2) a static field of a resident package P contains a reference to O (O is reachable from P),**
- **(3) there exists a valid remote reference to O (O is remote reachable)[10],**
- **(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): S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if, (1) S.ADEL is currently selected, (2) there is no instance in the context of O.APPLET that is active in any logical channel and (3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE22], §8.5) O.JAVAOBJECT is remote[10] reachable.**
- **R.JAVA.15 ([JCRE22], §11.3.4.1, Multiple Applet Instance Deletion): S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if, (1) S.ADEL is currently selected, (2) there is no instance of any of the O.APPLET being deleted that is active in any logical channel and (3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE22], §8.5) O.JAVAOBJECT is remote[10] reachable.**
- **R.JAVA.16 ([JCRE22], §11.3.4.2, Applet/Library Package Deletion): S.ADEL may perform OP.DELETE_PCKG upon an O.CODE_PKG only if, (1) S.ADEL is currently selected, (2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PKG that is an instance of a class that belongs to O.CODE_PKG, exists on the card and (3) there is no resident package on the card that depends on O.CODE_PKG.**
- **R.JAVA.17 ([JCRE22], §11.3.4.3, Applet Package and Contained Instances Deletion): S.ADEL may perform OP.DELETE_PCKG_APPLET upon an O.CODE_PKG only if, (1) S.ADEL is currently selected, (2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PKG, which is an instance of a class**

[10] Application note by the ST author: This requirement is irrelevant for the TOE because RMI is not supported.

that belongs to O.CODE_PKG exists on the card, (3) there is no package loaded on the card that depends on O.CODE_PKG, and (4) for every O.APPLET of those being deleted it holds that: (i) there is no instance in the context of O.APPLET that is active in any logical channel and (ii) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted, or ([JCRE22], §8.5) O.JAVAOBJECT is remote[10] reachable.

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

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

Application note:

FDP_ACF.1.2/ADEL:

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

FDP_RIP.1/ADEL Subset residual information protection

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

Application note:

Deleted freed resources (both code and data) 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.

FMT_MSA.1/ADEL Management of security attributes

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

FMT_MSA.3/ADEL Static attribute initialisation

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

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

FMT_SMF.1/ADEL Specification of Management Functions

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

FMT_SMR.1/ADEL Security roles

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

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

FPT_FLS.1/ADEL Failure with preservation of secure state

FPT_FLS.1.1/ADEL The TSF shall preserve a secure state when the following types of failures occur: **the applet deletion manager fails to delete a package/applet as described in** [JCRE22]**, §11.3.4**.

Application note:

- The TOE provides additional feedback information to the card manager in case of a potential security violation (see FAU_ARP.1).
- The Package/applet instance deletion is atomic. The "secure state" referred to in the requirement complies with Java Card specification ([JCRE22], §11.3.4.)

8.1.4 ODELG SECURITY FUNCTIONAL REQUIREMENTS

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

FDP_RIP.1/ODEL Subset residual information protection

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

Application note:

- Freed data resources resulting from the invocation of the method javacard.framework.JCSystem.requestObjectDeletion() is reused. Requirements on deallocation after the invocation of the method are described in **[JCAPI304]**.
- There is no conflict with FDP_ROL.1 here because of the bounds on the rollback mechanism: the execution of requestObjectDeletion() is not in the scope of the rollback because it must be performed in between APDU command processing, and therefore no transaction are in progress.

FPT_FLS.1/ODEL Failure with preservation of secure state

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

Application note:

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

8.1.5 CARG SECURITY FUNCTIONAL REQUIREMENTS

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

FCO_NRO.2/CM Enforced proof of origin

FCO_NRO.2.1/CM The TSF shall enforce the generation of evidence of origin for transmitted **application packages** at all times.

Application Note:

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

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

FCO_NRO.2.3/CM The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **by the DAP verification result.**

Application Note:

For this TOE the card manager performs an immediate verification of the origin of the package using an electronic signature mechanism, and no evidence is kept on the card for future verifications.

FDP_IFC.2/CM Complete information flow control

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

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

Application note:

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

FDP_IFF.1/CM Simple security attributes

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

(1) The keys used by the subjects S.INSTALLER and S.CARDMANAGER acting on behalf of the card issuer to decrypt and verify received messages;

(2) Authentication retry counter.

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

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

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

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

FDP_IFF.1.4/CM The TSF shall explicitly authorise an information flow based on the following rules: **The information flow is authorised according the relevant rules in Appendix E of** [GP221]**.**

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

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

Application note:

FDP_IFF.1.1/CM:

- The security attributes used to enforce the PACKAGE LOADING SFP depend on the communication protocol enforced between the CAD and the card. For instance, some of the attributes that are used are: (1) the keys used by the subjects to encrypt/decrypt their messages; (2) the number of pieces the application package has been split into in order to be sent to the card; (3) the ordinal of each piece in the decomposition of the package, etc. See for example Appendix D of [GP221].

FDP_IFF.1.2/CM:

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

FDP_IFF.1.5/CM

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

FDP_UIT.1/CM Data exchange integrity

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

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

Application note:

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

FIA_UID.1/CM Timing of identification

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.

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

Application note:

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

FMT_MSA.1/CM Management of security attributes

FMT_MSA.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to restrict the ability to **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/CM Static attribute initialisation

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

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

FMT_SMF.1/CM Specification of Management Functions

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

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

FMT_SMR.1/CM Security roles

FMT_SMR.1.1/CM The TSF shall maintain the roles: **the installer, the card acceptance device.**

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

FTP_ITC.1/CM Inter-TSF trusted channel

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

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

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

Application note:

There is no dynamic package loading on the Java Card platform. New packages are installed on the card only on demand of the card issuer.

8.1.6 CMGR Security Functional Requirements

In the PP [JCSPP], objectives for Card Management were objectives for the environment. Since the card manager has been defined to be part of the TOE, they were transformed into objectives for the TOE and have to be covered by SFRs.

FTP_ITC.1/CMGR Inter-TSF trusted channel

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

FTP_ITC.1.2/CMGR
The TSF shall permit **another trusted IT product** to initiate communication via the trusted channel.

FTP_ITC.1.3/CMGR
The TSF shall initiate communication via the trusted channel for **package loading, applet installation, applet deletion or package deletion**.

8.1.7 SCPG Security Functional Requirements

In the PP [JCSPP], objectives for the smart card platform are objectives for the environment. Since the smart card platform has been defined to be part of the TOE, they were transformed into objectives for the TOE and have to be covered by SFRs.

FPT_PHP.3 Resistance to physical attack

FPT_PHP.3.1

The TSF shall resist **physical manipulation and physical probing**[11] to the **TSF** by responding automatically such that the SFRs are always enforced.

8.2 Security Assurance Requirements

The security assurance requirement level is EAL5 augmented with ALC_DVS.2 and AVA_VAN.5.

8.3 Security Requirements Rationale

8.3.1 Objectives

Those parts of the rationale which deviate from the rationale in the PP [JCSPP] are underlined for easier comparison.

8.3.1.1 Security Objectives for the TOE

Application note by the ST author: In the PP, for some objectives explanations are added about coverages in the case when the TOE provides JCRMI functionality. Since the TOE does not provide RMI, these sentences have been omitted for brevity.

8.3.1.1.1 Identification

O.SID Subjects' identity is AID-based (applets, packages), and is met by the following SFRs:

FDP_ITC.2/Installer, FIA_ATD.1/AID, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_MSA.1/ADEL, FMT_MSA.1/CM, FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.3/CM, FMT_SMF.1/CM, FMT_SMF.1/ADEL, FMT_MTD.1/JCRE and FMT_MTD.3/JCRE.

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

8.3.1.1.2 Execution

O.FIREWALL This objective is met by the FIREWALL access control policy FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL, the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) and the functional requirement FDP_ITC.2/Installer. The functional requirements of the class FMT

[11] As assigned in [IFX_STLite] and [PP0084].

(FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMF.1, FMT_SMR.1/ADEL, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_MSA.2/FIREWALL_JCVM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM) also indirectly contribute to meet this objective.

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

Protection of the array parameters of remotely invoked methods[12], which are global as well, is covered by the general initialization of method parameters (FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL and FDP_RIP.1/TRANSIENT).

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

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

O.OPERATE The TOE is protected in various ways against applets' actions (FPT_TDC.1), the FIREWALL access control policy FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL, and is able to detect and block various failures or security violations during usual working (FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FAU_ARP.1). Its security-critical parts

12 Application note by the ST author: This sentence is irrelevant because the TOE does not support RMI.

and procedures are also protected: safe recovery from failure is ensured (FPT_RCV.3/Installer), applets' installation is cleanly aborted (FDP_ROL.1/FIREWALL), communication with external users and their internal subjects is well-controlled (FDP_ITC.2/Installer, FIA_ATD.1/AID, FIA_USB.1/AID) to prevent alteration of TSF data (also protected by components of the FPT class).

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

Application note: Startup of the TOE (TSF-testing) is covered by FPT_TST.1. This SFR component is not mandatory in [JCRE22], but appears in most of security requirements documents for masked applications. Self-tests are mandatory in order to comply with FIPS certification [FIPS 140-2].

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

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

8.3.1.1.3 Services

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

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

O.KEY-MNGT This relies on the same security functional requirements as O.CIPHER, plus FDP_RIP.1 and FDP_SDI.2 as well. Precisely it is met by the following components:

FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FCS_COP.1, FPR_UNO.1, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL and FDP_RIP.1/TRANSIENT.

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

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

8.3.1.1.4 Object Deletion

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

8.3.1.1.5 Applet Management

O.DELETION This security objective specifies that applet and package deletion must be secure. The non-introduction of security holes is ensured by the ADEL access control policy (FDP_ACC.2/ADEL, FDP_ACF.1/ADEL). The integrity and confidentiality of data that does not belong to the deleted applet or package is a by-product of this policy as well.

Non-accessibility of deleted data is met by FDP_RIP.1/ADEL and the TSFs are protected against possible failures of the deletion procedures (FPT_FLS.1/ADEL, FPT_RCV.3/Installer). The security functional requirements of the class FMT (FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL) included in the group ADELG also contribute to meet this objective.

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

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

O.CARD-MANAGEMENT This security objective specifies that the access to card management functions is secure.

The objective is met by the SFRs of the Card Management Group (see 8.1.6) (FTP_ITC.1.1 – FTP_ITC.1.3 /CMGR).

8.3.1.1.6 Smart card platform

O.SCP.IC This security objective is covered by FPT_PHP.3 (resistance against physical attacks).

O.SCP.RECOVERY This security objective is covered by FPT_FLS.1 and FAU_ARP.1. FPT_FLS.1 states that the TOE shall preserve a secure state in those cases defined in FAU_ARP.1, one of which refers to card tearing and power failure.

O.SCP.SUPPORT These objectives are covered as follows: Non-bypassability by FDP_SDI.2 (because data are secured against modification), low-level-cryptographic support by FCS_COP.1 and low-level transaction mechanism by FDP_ROL.1/FIREWALL (because it makes the operation OP.JAVA atomic). Non-bypassability and memory domain separation shall be investigated in ADV_ARC as of CC version 3.

8.3.2 Rationale Tables of Security Objectives and SFRs

Security Objectives	Security Functional Requirements	Rationale
O.SID	FIA_ATD.1/AID, FIA_UID.2/AID, FMT_MSA.1/JCRE, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.1/CM,	Section 8.3.1.1.1

	FMT_MSA.3/CM, FDP_ITC.2/Installer, FMT_SMF.1/CM, FMT_SMF.1/ADEL, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FIA_USB.1/AID, FMT_MSA.1/JCVM, FMT_MSA.3/JCVM	
O.FIREWALL	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FMT_SMR.1/Installer, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_MSA.3/FIREWALL, FMT_SMR.1, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FMT_MSA.1/JCRE, FDP_ITC.2/Installer, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_SMF.1, FMT_MSA.2/FIREWALL_JCVM, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_MSA.1/JCVM, FMT_MSA.3/JCVM	Section 8.3.1.1.2
O.GLOBAL_ARRAYS _CONFID	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FDP_RIP.1/bArray, FDP_RIP.1/APDU, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT	Section 8.3.1.1.2
O. LOBAL_ARRAYS _INTEG	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM	Section 8.3.1.1.2
O.NATIVE	FDP_ACF.1/FIREWALL	Section 8.3.1.1.2
O.OPERATE	FAU_ARP.1, FDP_ROL.1/FIREWALL, FIA_ATD.1/AID, FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FDP_ITC.2/Installer, FPT_RCV.3/Installer, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FPT_TDC.1, FIA_USB.1/AID, FPT_TST.1	Section 8.3.1.1.2
O.REALLOCATION	FDP_RIP.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS,	Section 8.3.1.1.2

	FDP_RIP.1/TRANSIENT, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS	
O.RESOURCES	FAU_ARP.1, FDP_ROL.1/FIREWALL, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMR.1/ADEL, FPT_FLS.1/Installer, FPT_FLS.1/ODEL, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_RCV.3/Installer, FMT_SMR.1/CM, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_SMF.1, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE	Section 8.3.1.1.2
O.ALARM	FPT_FLS.1/Installer, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL, FAU_ARP.1	Section 8.3.1.1.3
O.CIPHER	FCS_CKM.1.1/RSA, FCS_CKM.1.1/ECC, FCS_CKM.1.1/3DES, FCS_CKM.1.1/AES, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FCS_COP.1.1/RSA-CRT-SIGN, FCS_COP.1.1/RSA-SIGN, FCS_COP.1.1/RSA-VERI, FCS_COP.1.1/MAC-DES, FCS_COP.1.1/MAC-AES, FCS_COP.1.1/CMAC-AES, FCS_COP.1.1/3DES, FCS_COP.1.1/AES, FCS_COP.1.1/RSA-DEC, FCS_COP.1.1/RSA-CRT-DEC, FCS_COP.1.1/RSA-ENC, FCS_COP.1.1/ECDSA-SIGN, FCS_COP.1.1/ECDSA-VERI, FPR_UNO.1, FCS_RNG.1, FCS_COP.1.1/HASH	Section 8.3.1.1.3
O.KEY-MNGT	FCS_CKM.1.1/RSA, FCS_CKM.1.1/ECC , FCS_CKM.1.1/3DES, FCS_CKM.1.1/AES, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FCS_COP.1.1/RSA-CRT-SIGN, FCS_COP.1.1/RSA-SIGN, FCS_COP.1.1/RSA-VERI, FCS_COP.1.1/MAC-DES, FCS_COP.1.1/MAC-AES, FCS_COP.1.1/CMAC-AES, FCS_COP.1.1/3DES, FCS_COP.1.1/AES, FCS_COP.1.1/RSA-DEC, FCS_COP.1.1/RSA-CRT-DEC, FCS_COP.1.1/RSA-ENC,	Section 8.3.1.1.3

	FCS_COP.1.1/ECDSA-SIGN, FCS_COP.1.1/ECDSA-VERI, FPR_UNO.1, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_SDI.2, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FCS_COP.1.1/HASH	
O.PIN-MNGT	FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FPR_UNO.1, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FDP_ROL.1/FIREWALL, FDP_SDI.2, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL	Section 8.3.1.1.3
O.TRANSACTION	FDP_ROL.1/FIREWALL, FDP_RIP.1/ABORT, FDP_RIP.1/ODEL, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FDP_RIP.1/OBJECTS	Section 8.3.1.1.3
O.OBJ-DELETION	FDP_RIP.1/ODEL, FPT_FLS.1/ODEL	Section 8.3.1.1.4
O.DELETION	FDP_ACC.2/ADEL, FDP_ACF.1/ADEL, FDP_RIP.1/ADEL, FPT_FLS.1/ADEL, FPT_RCV.3/Installer, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL	Section 8.3.1.1.5
O.LOAD	FCO_NRO.2/CM, FDP_IFC.2/CM, FDP_IFF.1/CM, FDP_UIT.1/CM, FIA_UID.1/CM, FTP_ITC.1/CM	Section 8.3.1.1.5
O.INSTALL	FDP_ITC.2/Installer, FPT_RCV.3/Installer, FPT_FLS.1/Installer	Section 8.3.1.1.5
O.CARD-MANAGEMENT	FTP_ITC.1/CMGR	Section 8.3.1.1.5
O.SCP.IC	FPT_PHP.3	Section 8.3.1.1.6
O.SCP.RECOVERY	FPT_FLS.1, FAU_ARP.1	Section

		8.3.1.1.6
O.SCP.SUPPORT	FCS_COP.1.1/RSA-CRT-SIGN, FCS_COP.1.1/RSA-SIGN, FCS_COP.1.1/RSA-VERI, FCS_COP.1.1/MAC-DES, FCS_COP.1.1/MAC-AES, FCS_COP.1.1/CMAC-AES, FCS_COP.1.1/3DES, FCS_COP.1.1/AES, FCS_COP.1.1/RSA-DEC, FCS_COP.1.1/RSA-CRT-DEC, FCS_COP.1.1/RSA-ENC, FCS_COP.1.1/ECDSA-SIGN, FCS_COP.1.1/ECDSA-VERI, FDP_ROL.1/FIREWALL, FDP_SDI.2, FCS_COP.1.1/HASH	Section 8.3.1.1.6

Table 7: Security Objectives and SFRs – Coverage

Security Functional Requirements	Security Objectives
FDP_ACC.2/FIREWALL	O.FIREWALL, O.OPERATE, O.PIN-MNGT
FDP_ACF.1/FIREWALL	O.FIREWALL, O.NATIVE, O.OPERATE, O.PIN-MNGT
FDP_IFC.1/JCVM	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG
FDP_IFF.1/JCVM	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG
FDP_RIP.1/OBJECTS	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FMT_MSA.1/JCRE	O.SID, O.FIREWALL
FMT_MSA.1/JCVM	O.SID, O.FIREWALL
FMT_MSA.2/FIREWALL_JCVM	O.FIREWALL
FMT_MSA.3/FIREWALL	O.SID, O.FIREWALL
FMT_MSA.3/JCVM	O.SID, O.FIREWALL
FMT_SMF.1	O.FIREWALL, O.RESOURCES
FMT_SMR.1	O.FIREWALL, O.RESOURCES
FCS_CKM.1.1/RSA, FCS_CKM.1.1/ECC,	O.CIPHER, O.KEY-MNGT

FCS_CKM.1.1/3DES, FCS_CKM.1.1/AES	
FCS_CKM.2	O.CIPHER, O.KEY-MNGT
FCS_CKM.3	O.CIPHER, O.KEY-MNGT
FCS_CKM.4	O.CIPHER, O.KEY-MNGT
FCS_COP.1.1/RSA-CRT-SIGN, FCS_COP.1.1/RSA-SIGN, FCS_COP.1.1/RSA-VERI, FCS_COP.1.1/MAC-DES, FCS_COP.1.1/MAC-AES, FCS_COP.1.1/CMAC-AES, FCS_COP.1.1/3DES, FCS_COP.1.1/AES, FCS_COP.1.1/RSA-DEC, FCS_COP.1.1/RSA-CRT-DEC, FCS_COP.1.1/RSA-ENC, FCS_COP.1.1/ECDSA-SIGN, FCS_COP.1.1/ECDSA-VERI, FCS_COP.1.1/HASH	O.CIPHER, O.KEY-MNGT, O.SCP.SUPPORT
FCS_RNG.1	O.CIPHER
FDP_RIP.1/ABORT	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FDP_RIP.1/APDU	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FDP_RIP.1/bArray	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FDP_RIP.1/KEYS	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FDP_RIP.1/TRANSIENT	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION
FDP_ROL.1/FIREWALL	O.OPERATE, O.RESOURCES, O.PINMNGT, O.TRANSACTION, O.SCP.SUPPORT
FAU_ARP.1	O.OPERATE, O.RESOURCES, O.ALARM,

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

FCO_NRO.2/CM	O.LOAD
FDP_IFC.2/CM	O.LOAD
FDP_IFF.1/CM	O.LOAD
FDP_UIT.1/CM	O.LOAD
FIA_UID.1/CM	O.LOAD
FMT_MSA.1/CM	O.SID, O.FIREWALL
FMT_MSA.3/CM	O.SID, O.FIREWALL
FMT_SMF.1/CM	O.SID, O.FIREWALL, O.RESOURCES
FMT_SMR.1/CM	O.FIREWALL, O.RESOURCES
FTP_ITC.1/CM	O.LOAD
FTP_ITC.1/CMGR	O.CARD-MANAGEMENT
FPT_PHP.3	O.SCP.IC

Table 8: SRFs and Security Objectives

8.3.3 SFR Dependencies

The SFRs are listed in the same order as in chapter 8.1. An SFR can appear more than once since there are different groups. If an SFR has no dependencies, it is listed only once in the table (even if it applies to more than one group).

Unsatisfied dependencies are marked in **bold** and justified below.

SFR	Dependencies	Satisfied Dependencies
FDP_ACC.2/ FIREWALL	(FDP_ACF.1)	FDP_ACF.1/ FIREWALL
FDP_ACF.1/ FIREWALL	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.2/FIREWALL, FMT_MSA.3/FIREWALL
FDP_IFC.1/JCVM	(FDP_IFF.1)	FDP_IFF.1/JCVM
FDP_IFF.1/JCVM	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.1/JCVM, FMT_MSA.3/JCVM
FDP_RIP.1	No Dependencies	-
FMT_MSA.1/JCRE	(FDP_ACC.1 or FDP_IFC.1) and (**FMT_SMF.1**) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FMT_SMR.1
FMT_MSA.2/Firewall_JCVM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM,

SFR	Dependencies	Satisfied Dependencies
		FMT_SMR.1
FMT_MSA.3/Firewall	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1
FMT_SMF.1	No Dependencies	-
FMT_SMF.1/ADEL	No Dependencies	-
FMT_SMR.1	(FIA_UID.1)	FIA_UID.2/AID
FCS_CKM.1.1/RSA, FCS_CKM.1.1/ECC, FCS_CKM.1.1/3DES, FCS_CKM.1.1/AES	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_CKM.2, FCS_CKM.4
FCS_CKM.2	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1.1/RSA, FCS_CKM.1.1/ECC, FCS_CKM.1.1/3DES, FCS_CKM.1.1/AES, FCS_CKM.4
FCS_CKM.3	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1.1/RSA, FCS_CKM.1.1/ECC, FCS_CKM.1.1/3DES, FCS_CKM.1.1/AES FCS_CKM.4
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	FCS_CKM.1.1/RSA, FCS_CKM.1.1/ECC, FCS_CKM.1.1/3DES, FCS_CKM.1.1/AES
FCS_COP.1.1/RSA-CRT-SIGN, FCS_COP.1. 1/RSA-SIGN, FCS_COP.1.1/RSA-VERI, FCS_COP.1.1/RSA-DEC, FCS_COP.1.1/RSA-CRT-DEC, FCS_COP.1.1/RSA-	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1.1/RSA, FCS_CKM.1.1/ECC , FCS_CKM.4

SFR	Dependencies	Satisfied Dependencies
ENC, FCS_COP.1.1/ECDSA-SIGN, FCS_COP.1.1/ ECDSA-VERI		
FCS_COP.1.1/CMAC-AES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1.1/AES, FCS_CKM.4
FCS_COP.1.1/MAC-AES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1.1/AES, FCS_CKM.4
FCS_COP.1.1/AES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1.1/AES, FCS_CKM.4
FCS_COP.1.1/MAC-DES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1.1/3DES, FCS_CKM.4
FCS_COP.1.1/3DES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1.1/3DES, FCS_CKM.4
FCS_COP.1.1/HASH	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	Non-satisfied dependency because no Key generation and destruction for Hash necessary.
FCS_RNG.1	No Dependencies	-
FDP_ROL.1/Firewall	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM
FAU_ARP.1	**(FAU_SAA.1)**	
FDP_SDI.2	No Dependencies	-
FPR_UNO.1	No Dependencies	-
FPT_FLS.1	No Dependencies	-
FPT_FLS.1/ADEL	No Dependencies	-

SFR	Dependencies	Satisfied Dependencies
FPT_TDC.1	No Dependencies	-
FPT_TST.1	No Dependencies	-
FIA_ATD.1/AID	No Dependencies	-
FIA_UID.2/AID	No Dependencies	-
FIA_USB.1/AID	(FIA_ATD.1)	FIA_ATD.1/AID
FMT_MTD.1/JCRE	(FMT_SMF.1) and (FMT_SMR.1)	FMT_SMF.1, FMT_SMR.1
FMT_MTD.3/JCRE	(FMT_MTD.1)	FMT_MTD.1/JCRE
FDP_ITC.2/Installer	(FDP_ACC.1 or FDP_IFC.1) and (FPT_TDC.1) and (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM, FPT_TDC.1
FMT_SMR.1/Installer	**(FIA_UID.1)**	
FPT_RCV.3/Installer	(AGD_OPE.1)	AGD_OPE.1
FDP_ACC.2/ADEL	(FDP_ACF.1)	FDP_ACF.1/ADEL
FDP_ACF.1/ADEL	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.2/ADEL, FMT_MSA.3/ADEL
FMT_MSA.1/ADEL	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/ADEL, FMT_SMF.1/ADEL, FMT_SMR.1/ADEL
FMT_MSA.3/ADEL	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/ADEL, FMT_SMR.1/ADEL
FMT_SMR.1/ADEL	**(FIA_UID.1)**	
FCO_NRO.2/CM	(FIA_UID.1)	FIA_UID.1/CM
FDP_IFC.2/CM	(FDP_IFF.1)	FDP_IFF.1/CM
FDP_IFF.1/CM	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/CM, FMT_MSA.3/CM
FDP_UIT.1/CM	(FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM
FIA_UID.1/CM	No Dependencies	-
FMT_MSA.1/CM	(FDP_ACC.1 or	FDP_IFC.2/CM,

SFR	Dependencies	Satisfied Dependencies
	FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FMT_SMF.1/CM, FMT_SMR.1/CM
FMT_MSA.1/JCVM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_SMF.1, FMT_SMR.1
FMT_MSA.3/CM	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/CM, FMT_SMR.1/CM
FMT_MSA.3/JCVM	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/JCVM, FMT_SMR.1
FMT_SMF.1/CM	No Dependencies	-
FMT_SMR.1/CM	(FIA_UID.1)	FIA_UID.1/CM
FTP_ITC.1/CM	No Dependencies	-
FTP_ITC.1/CMGR	No Dependencies	-
FPT_PHP.3	No Dependencies	-

Table 9: SFRs dependencies

The dependency FIA_UID.1 of FMT_SMR.1/Installer is discarded.

This ST does not require the identification of the "installer" since it is considered as part of the TSF.

The dependency FIA_UID.1 of FMT_SMR.1/ADEL is discarded.

This ST does not require the identification of the "deletion manager" since it is considered as part of the TSF.

The dependency FMT_SMF.1 of FMT_MSA.1/JCRE is discarded.

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

The dependency FAU_SAA.1 of FAU_ARP.1 is discarded.

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

8.3.4 Security assurance requirement dependencies

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

Requirements	CC Dependencies	Satisfied Dependencies
	(ATE_COV.1) and (ATE_FUN.1)	ATE_COV.2, ATE_FUN.1
AVA_VAN.5	(ADV_ARC.1) and (ADV_FSP.4) and (ADV_IMP.1) and (ADV_TDS.3) and (AGD_OPE.1) and (AGD_PRE.1) and (ATE_DPT.1)	ADV_ARC.1, ADV_FSP.5, ADV_IMP.1, ADV_TDS.4, AGD_OPE.1, AGD_PRE.1, ATE_DPT.3

Table 10 SARs Dependencies

8.3.5 Rationale for the Security Assurance Requirements

EAL5 is required for this TOE and product since it is intended to defend against highly sophisticated attacks.

This evaluation assurance level permits a developer to gain maximum assurance from security engineering based upon rigorous commercial development practises supported by moderate application of specialist security engineering techniques.

In order to provide the evaluator a meaningful level of assurance that the TOE and its embedded product provide a high level of independently assured security in a planned development this requires a rigorous development approach without incurring unreasonable costs attributable to specialist security engineering techniques: the evaluator should have access to a modular TSF design and semiformal design description, a more structured (and hence analysable) architecture. The lowest EAL level for which such access is required is EAL5.

The reason for going beyond the EAL 4+ level as given in the JCS PP [JCSPP] is that the TOE is planned as the underlying platform for future security sensitive government applications that need a trustworthy foundation intended to defend against highly sophisticated attacks.

The chosen augmentation with ALC_DVS.2 and AVA_VAN.5 components increases the assurance level additionally.

For ALC_DVS.2 the sufficiency of the security measures has to be justified by the TOE developer to provide the necessary level of protection to maintain the confidentiality and integrity of the TOE which is especially necessary for sensitive government applications.

For AVA_VAN.5 an advanced methodical vulnerability analysis of the TOE has to be performed to confirm that the potential vulnerabilities cannot be exploited in the operational environment for the TOE. Penetration testing is performed by the evaluator assuming an attack potential of High that also adds assurance for sensitive government applications.

9 TOE summary specification

9.1 TOE Security functions

9.1.1 SF.TRANSACTION

This security function provides atomic transactions according to the Java Card Transaction and Atomicity mechanism with commit and rollback capability ([JCRE304], Section 7)[13] for updating persistent data in FLASH memory.

The update operation either successfully completes or the data is restored to its original pre-transaction state if the transaction does not complete normally. The rollback operation restores the original values of the persistent data and clears the dedicated transaction area. The TOE permits rollback of any access in the sense of [JCRE304], Section 6.2.8, and creation of objects via the JCAPI new or makeTransient calls.

9.1.2 SF.ACCESS_CONTROL

This security function provides control for the TOE. It is in charge of the FIREWALL access control SFP and the JCVM information flow control SFP.
The Firewall access control policy and the JCVM information flow control policy are enforced at runtime. It defines how accessing the following items: Static Class Fields, Array Objects, Class Instance Object Fields, Class Instance Object Methods, Standard Interface Methods, Shareable Interface Methods, Classes, Standard Interfaces, Shareable Interfaces, Array Object Methods.

Based on security attributes [Sharing, Context, Lifetime], it performs access control to object fields between objects and throws security exception when access is denied. It enforces applet isolation located in different packages and controls the access to global data containers shared by all applet instances.

The JCRE allocates and manages a context for each Java API package containing applets. The JCRE maintains its own context as a special system privilege so that it can perform operations that are denied to contexts of applets.

1. The TOE enforces the Firewall access control SFP and the JCVM information flow policy to control the flow of information between subjects.
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

[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.

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. For the TOE every action is always performed by an identified user: the currently selected applet or the package that is the subject's owner. Means of identification are provided during the loading procedure of the package and the registration of applet instances. The TOE requires each of the above stated users to identify itself before allowing any other TSF-mediated actions on behalf of that user and associates the following user security attributes with subjects (like the package) acting on behalf of that user: Package AID.
4. The TOE accepts only secure values for security attributes.
5. The ability to modify the Currently Active Context and the Active Applets is restricted to the Java Card VM (S.JCVM). The ability to modify the Selected Applet Context is restricted to the Java Card RE (S.JCRE).
6. 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.

9.1.3 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 RSA-CRT and RSA according to [JCAPI304]; [AIS20] DRG.4 and ECC according to [JCAPI304] and 3DES according to [FIPS46-3] and AES according to [FIPS 197].
    - Key generation refers to the generation of a cryptographic key or key pair to be used in cryptographic algorithms. The algorithms supported by the TOE that require a secret or private key are RSA-CRT, RSA, ECC, Triple-DES and AES. Key generation involves generation of a secret value that is used as a secret key for a symmetric algorithm (AES or Triple-DES), or a private key for an asymmetric algorithm (DSA, ECDSA, ECDH), or a prime generation seed for RSA.
    - Key access and distribution: the TOE provides 3-DES key (112, 168 bit), RSA (512 up to 2048 bit), RSA-CRT (512 up to 4096 bit), ECC (160, 192, 224, 256, 384, 512 and 521 bit) and AES (128, 192, 256 bit) access and distribution in accordance with [JCAPI304]. Key access is provided via the Java Card API get methods of classes AESKey, DESKey, ECKey, ECPrivateKey, ECPublicKey, RSAPrivateCrtKey, RSAPrivateKey and RSAPublicKey.

- Key distribution is provided via the Java Card API set methods of javacard.security classes AESKey, DESKey, ECKey, ECPrivateKey, ECPublicKey, RSAPrivateCrtKey, RSAPrivateKey and RSAPublicKey

2. Key destruction: The TOE provides key destruction for 3-DES, AES, RSA, RSA-CRT and ECDSA keys by the following means:
   - Applications may use the Java Card API method Key.clearKey() for key destruction.
   - An authenticated off-card entity may use the PUT KEY command within a Secure Channel Session to zeroize the DAP key(s) and the Delegated Management Token and Receipt keys.
   - All keys (and the Global PIN) are zeroized by setting the Issuer Security Domain life cycle state to TERMINATED. An authenticated off-card entity may use the SET STATUS command for this purpose.
   - The TOE zeroizes cryptographic session keys when closing the Secure Channel Session or upon card reset.
   - In order to delete the DAP Verification key the Security Domain containing this key must be deleted. This operation deletes all keys contained in that Security Domain.
3. Encryption/decryption and sign/verify in accordance with a specified cryptographic algorithm 3-DES in CBC/ ECB mode, RSA, RSA-CRT, ECC and AES in CBC/ECB mode.
   - Encryption and decryption with Triple-DES and AES in CBC and ECB modes, RSA and RSA-CRT is provided via the Java Card API methods. AES is implemented according to [FIPS 197], Triple-DES according to [SP800-67], the CBC and ECB modes of operation according to [SP800-38a], and the RSA cipher according to [PKCS1]. Triple-DES is used for Secure Channel and sensitive data encryption and decryption according to [GP221], E.4.6 and E.4.7.
   - Digital signature generation and verification using RSA, RSA-CRT and ECDSA is provided to applications via the Java Card API methods defined in the javacard.security.Signature class. The implementation of the algorithm is according to [PKCS1] for RSASSA-PKCS1-v1_5 and RSASSA-PSS, and [FIPS 186-3] for ECDSA.
4. Hash calculation according to SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512.
   - Applications may use the methods of the Java Card API class javacard.security.MessageDigest for hashing with SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512. A Security Domain uses SHA-1 for

Load File Data Block Hash generation. The Hash algorithms are implemented according to [FIPS 180-3].

5. MAC generation and verification in accordance with a specified cryptographic algorithm DES CBC-MAC, DES Retail-MAC, AES CBC-MAC and AES CMAC.
    - MAC generation with Triple-DES in CBC mode and AES MAC is supported via the Java Card API javacard.security.Signature.
      Full Triple-DES MAC generation according to [ISO9797-1] is used for authentication cryptogram generation and verification for the GlobalPlatform Secure Channel Protocol [GP221], E.4.2, B.1.2.1) and [GP AM D].
6. Random number generation that meet DRG.4 according [AIS20].
    - The random number generator provided by the TOE is a deterministic random bit generator based on the AES block cipher according to [SP800-90A] that meets DRG.4 of [AIS20]. Besides its use in key generation, applications may use the methods of the Java Card API javacard.security.RandomData class for generation of random numbers.

9.1.4 SF.INTEGRITY

This security function provides a means to check the integrity of checksummed data stored in FLASH memory.

The security function provides means to securely manage operations associated with sensitive data like keys and PINs by checking the integrity of the data stored in it by cyclic redundancy checks (reed solomon code).

1. This security function initializes the checksum of cryptographic keys, PIN values and their associated security attributes.
2. The TOE monitors cryptographic keys, PIN values and their associated security attributes stored within the TSF 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/or switch to an endless loop and therefore prevent the usage of this key/PIN. This is a secure state.
4. The TOE runs packages checksum integrity tests during initial start-up at each power on of the TOE.

9.1.5 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.

The TSF provides the preservation of a secure state by managing security violations thus resulting in an immediate reset.

The TSF ensures resistance to physical tampering using features against probing and an active shield detecting integrity violation.

The security function ensures that sensitive data are locked upon the following operations as defined in [JCRE304]:

- Deletion of package and/or applications,
- Deletion of objects.

They are erased upon deallocation of the objects.

This security function also ensures that the sensitive temporary buffers (transient object, bArray object, APDU buffer, Cryptographic buffer) are securely cleared after their usage with respect to their life-cycle and interface as defined in [JCRE304]. Transient objects and persistent objects are erased upon deallocation of the object.

The TSF ensures resistance to physical tampering using features against probing and an active shield detecting integrity violation.

1. The TOE throws an exception, locks the card, the application or 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.
2. The TOE ensures that an attacker is unable to observe cryptographic operations / comparison operations on key values / PIN values.
3. The TOE ensures that any previous information content of a resource is made unavailable upon deallocation of the resource from the bArray object, any reference to an object instance created during an aborted transaction and the cryptographic buffer. At least upon allocation of the APDU buffer any previous information content is made unavailable.
4. The TOE detects physical tampering of the TSF with sensors for operating voltage, clock frequency, temperature and electromagnetic radiation. It is resistant to physical tampering of the TSF. If the TOE detects with the above mentioned sensors that it is not supplied within the specified limits, a security reset is initiated and the TOE is not operable until the supply is back in the specified limits. The design of the hardware protects it against analysing and physical tampering.
5. The TOE hides information about IC power consumptions and command execution time, to ensure that no confidential information can be derived from this data.

9.1.6 SF.APPLET

This security function ensures the secure loading of a *package* or installing of an *applet by S.CAD and the secure* deletion of *applet*s and/or *packages by S.ADEL.*

Content management is the capability for the loading, installation, extradition, registry update and card content removal. These operations are performed by a privileged Security Domain that applies a secure communication policy. Secure communication is provided by the security function SF.CARRIER.

Content changes are permitted according to the privileges that have been assigned to the acting Security Domain.

Management of Security Domains is supported according to the GlobalPlatform specifications [GP CIC], [GP221], [GP ID_Config] and [GP SE_Config]. The TOE supports the management functions listed in FMT_SMF.1/CM.

1. When importing user data by loading of a *package* or installing of an *applet* e.g. the TOE enforces the evidence of the origin and the integrity of the corresponding data by appropriate identification and transmission mechanisms.
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]**,** §11.1.4. and enters a maintenance mode where the ability to return the TOE to a secure state is provided for reset, insufficient FLASH memory, failure in cryptographic safeguarding, package references (versions) mismatching
5. The TOE ensures the safe deletion of applets and/or packages.
6. The TOE restricts the ability to modify the Registered Applets and Resident Packages to the JCRE.
7. The TOE ensures that any previous information content of a resource is made unavailable upon the deallocation 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()` or if deletion operations according to ADEL access control SFP occur.

8. The TOE preserves a secure state when the applet deletion manager fails to delete a package/applet as described in [JCRE304], §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.

9.1.7 SF.CARRIER

This security function ensures secure downloading of applications on the card.

The TOE supports secure communication initiated by an off-card entity by the following means:

- Secure Channel Protocol 02 (SCP02) [GP221] provides the three followings levels of security: entity authentication, integrity and data origin authentication and confidentiality. A further level of security applies to sensitive data (e.g. secret keys) that shall always be transmitted as confidential data.
  SCP02 is realised by the TOE based on the 3-DES cryptographic algorithm (see also: 9.1.3, number 6).
- Secure Channel Protocol 03 (SCP03) [GP AM D] provides the three followings level of security: mutual authentication, integrity and data origin authentication and confidentiality. It is based on SCP02 and is a new secure channel protocol supporting AES-based cryptography.
  SCP03 is realised by the TOE based on the AES cryptographic algorithm (see also: FCS_CKM.1.1/AES, FCS_COP.1.1/AES and 9.1.3, number 6).

Applications can use the Secure Channel Protocol(s) supported by their associated Security Domain for securing information exchanged with the off-card entity.

The Secure Channel is used for the purpose of secure card content management that is covered by the security function SF.APPLET. Before performing card content management operations, the TOE checks if a Secure Channel Session has been successfully initiated.

Application selection, secure channel initiation, request data with the GET DATA command on behalf of the user can be performed before the user is identified.

The Secure Channel Protocol provides mutual authentication, integrity and data origin authentication and confidentiality of transmitted data
Mutual authentication is implemented by means of cryptographic exchange between the card and the off-card entity initiated by the off-card entity; it implies the generation of session keys derived from static key(s) maintained by the Security Domain. Message integrity and data origin authentication is assured by applying MAC calculation across the header and data field of an APDU command using the generated Secure Channel session MAC key. Confidentiality of message data is assured by encryption using the Secure Channel session ENC key.

The TOE provides capabilities to verify the source and the integrity of a particular block of code or data by means of Load File Data Block Hash (for verification of integrity) and Load File Data Block Signature (DAP authentication value) according

to [GP221], C.2 and C.3.
The DAP signature verification is realised by the TOE either with AES, RSA, ECC or 3-DES cryptography depending on the signature token created by the card issuer.
The TOE enforces the Secure Channel Protocol information flow control policy and rules, the Runtime behavior rules and Secure Channel behavior rules on the subjects S.CAD and S.SD involved in the exchange of messages.

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.INSTALLER, 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.
12. The TOE is capable of modifying the security attributes Card Life Cycle State and Security Level.

9.2 Assurance measures

This chapter describes the Assurance Measures fulfilling the requirements listed in chapter 8.

The following table lists the Assurance measures and references the corresponding documents describing the measures.

Assurance Measures	Description
AM_ADV	The representation of the TSF is described in the documentation for functional specification, in the documentation for TOE design, in the security architecture description and in the documentation for implementation representation.
AM_AGD	The guidance documentation is described in the operational guidance documentation and in the documentation for preparative procedures.
AM_ALC	The life cycle support of the TOE during its development and maintenance is described in the life cycle documentation including configuration management, delivery procedures, development security as well as development tools.
AM_ATE	The testing of the TOE is described in the test documentation.
AM_AVA	The evaluator uses the development and guidance documentation by the developer as a basis for his vulnerability analysis.

Table 11: Reference of Assurance Measures

9.3 Association tables of SFRs and TSS

Security Functional Requirements	TOE Summary Specification
FDP_ACC.2/FIREWALL	SF.ACCESS_CONTROL.1
FDP_ACF.1/FIREWALL	SF.ACCESS_CONTROL.1
FDP_IFC.1/JCVM	SF.ACCESS_CONTROL.1
FDP_IFF.1/JCVM	SF.ACCESS_CONTROL.1
FDP_RIP.1/OBJECTS	SF.SECURITY.3
FMT_MSA.1/JCRE	SF.ACCESS_CONTROL.5
FMT_MSA.1/JCVM	SF.ACCESS_CONTROL.5
FMT_MSA.2/FIREWALL_JCVM	SF.ACCESS_CONTROL.4
FMT_MSA.3/FIREWALL	SF.ACCESS_CONTROL.4
FMT_MSA.3/JCVM	SF.ACCESS_CONTROL.4
FMT_SMF.1	SF.ACCESS_CONTROL.2, 5
FMT_SMR.1	SF.ACCESS_CONTROL.1
FCS_CKM.1/RSA	SF.CRYPTO.1
FCS_CKM.1/ECC	SF.CRYPTO.1
FCS_CKM.1/3DES	SF.CRYPTO.1
FCS_CKM.1/AES	SF.CRYPTO.1
FCS_CKM.2	SF.CRYPTO.2
FCS_CKM.3	SF.CRYPTO.2
FCS_CKM.4	SF.CRYPTO.3
FCS_COP.1.1/RSA-CRT-SIGN	SF.CRYPTO.4
FCS_COP.1. 1/RSA-SIGN	SF.CRYPTO.4
FCS_COP.1.1/RSA-VERI	SF.CRYPTO.4
FCS_COP.1.1/MAC-DES	SF.CRYPTO.6
FCS_COP.1.1/MAC-AES	SF.CRYPTO.6
FCS_COP.1.1/CMAC-AES	SF.CRYPTO.6
FCS_COP.1.1/3DES	SF.CRYPTO.4
FCS_COP.1.1/AES	SF.CRYPTO.4
FCS_COP.1.1/RSA-DEC	SF.CRYPTO.4
FCS_COP.1.1/RSA-CRT-DEC	SF.CRYPTO.4
FCS_COP.1.1/RSA-ENC,	SF.CRYPTO.4
FCS_COP.1.1/ECDSA-SIGN	SF.CRYPTO.4
FCS_COP.1.1/ ECDSA-VERI	SF.CRYPTO.4

Security Functional Requirements	TOE Summary Specification
FCS_COP.1.1/HASH	SF.CRYPTO.5
FCS_RNG.1.1	SF.CRYPTO.7
FDP_RIP.1/ABORT	SF.SECURITY.3
FDP_RIP.1/APDU	SF.SECURITY.3
FDP_RIP.1/bArray	SF.SECURITY.3
FDP_RIP.1/KEYS	SF.SECURITY.3
FDP_RIP.1/TRANSIENT	SF.SECURITY.3
FDP_ROL.1/FIREWALL	SF.TRANSACTION
FAU_ARP.1	SF.SECURITY.1
FDP_SDI.2	SF.INTEGRITY.1, 2, 3
FPR_UNO.1	SF.SECURITY.2
FPT_FLS.1	SF.SECURITY.1
FPT_TDC.1	SF.ACCESS_CONTROL.6
FIA_ATD.1/AID	SF.ACCESS_CONTROL.2
FIA_UID.2/AID	SF.ACCESS_CONTROL.3
FIA_USB.1/AID	SF.ACCESS_CONTROL.1,2,3
FMT_MTD.1/JCRE	SF.ACCESS_CONTROL.5
FMT_MTD.3/JCRE	SF.ACCESS_CONTROL.4
FDP_ITC.2/Installer	SF.APPLET.1, 2, 3
FMT_SMR.1/Installer	SF.APPLET.1
FPT_FLS.1/Installer	SF.APPLET.4
FPT_RCV.3/Installer	SF.APPLET.4
FDP_ACC.2/ADEL	SF.APPLET.5
FDP_ACF.1/ADEL	SF.APPLET.5
FDP_RIP.1/ADEL	SF.APPLET.7
FMT_MSA.1/ADEL	SF.APPLET.6
FMT_MSA.3/ADEL	SF.APPLET.6
FMT_SMF.1/ADEL	SF.APPLET.6
FMT_SMR.1/ADEL	SF.APPLET.8
FPT_FLS.1/ADEL	SF.APPLET.8
FDP_RIP.1/ODEL	SF.APPLET.7
FPT_FLS.1/ODEL	SF.APPLET.8
FCO_NRO.2/CM	SF.CARRIER.1, 2, 3
FDP_IFC.2/CM	SF.CARRIER.6

Security Functional Requirements	TOE Summary Specification
FDP_IFF.1/CM	SF.CARRIER.6
FDP_UIT.1/CM	SF.CARRIER.6
FIA_UID.1/CM	SF.CARRIER.4, 5
FMT_MSA.1/CM	SF.CARRIER.6
FMT_MSA.3/CM	SF.CARRIER.7
FMT_SMF.1/CM	SF.CARRIER.12
FMT_SMR.1/CM	SF.CARRIER.8
FTP_ITC.1/CM	SF.CARRIER.9, 10, 11
FTP_ITC.1/CMGR	SF.CARRIER.9, 10, 11
FPT_PHP.3	SF.SECURITY.4, 5
FPT_TST.1	SF.INTEGRITY.4

Table 12: SFRs and TSS - Coverage

10 Statement of compatibility

This is a statement of compatibility between this Composite Security Target (Composite-ST) and the Platform Security Target (Platform-ST) of the Chip M5073 G11, [IFX_Cert] and [IFX_STLite]. This statement is compliant to the requirements of [SUPP].

10.1 Matching statement

The TOE relies on fulfilment of the following implicit assumptions on the IC:

- Certified IFX Microcontroller M5073 G11, [IFX_Cert], [IFX_STLite]
- True Random Number Generator (TRNG) according to AIS 31 [AIS31]

The rationale of the platform-ST has been used to identify the relevant SFRs, TOE objectives, threats and OSPs.

10.1.1 TOE Security Environment

10.1.1.1 Threats, OSPs and Assumptions

The only threat of this composite TOE which is directly related to IC functionality is T.PHYSICAL.

It can be mapped to the following platform-ST threats [IFX_STLite]:

- T.Phys_Manipulation
- T.Phys-Probing
- T.Malfunction
- T.Leak-Inherent
- T.Leak-Forced
- T.Abuse-Func
- T.Mem-Access

	T.Phys_Manipulation	T.Phys-Probing	T.Malfunction	T.Leak-Inherent	T.Leak-Forced	T.Abuse-Func	T.Mem-Access
T.PHYSICAL	x	x	x	x	x	x	x

Table 13: Mapping of threats

None of the OSPs of the composite TOE are applicable to the IC.

The OSPs from the platform-ST are as follows [IFX_STLite]:

- P.Process-TOE
- P.Add-Functions
- P.Crypto-Service
- P.Lim_Block_Loader

P.Add-Functions and P.Crypto-Service pertain to the supply of low-level cryptographic processing to the Java Card System which is covered by the security objective O.SCP.SUPPORT.
P.Lim_Block_Loader pertains to the composite Security IC Manufacturer or the TOE developer (see chapter 2.4.1, phase 4) who limits the capability and blocks the availability of the Loader in order to protect stored data from disclosure and manipulation.

OSP	**Classification of OSPs**	**Mapping to Security Objectives of this Composite-ST**
P.Process-TOE	not relevant	This OSP refers to identification of the single chips.
P.Add-Functions	relevant	O.SCP.SUPPORT
P.Crypto-Service	relevant	O.SCP.SUPPORT
P.Lim_Block_Loader	Not relevant	This OSP refers to organisational measures of the Security IC Manufacturer or the TOE developer

Table 14: Mapping of hardware OSPs to composite security objectives

The assumptions from this ST make no assumption on the platform but only on the environment of the TOE.

The assumptions from the Platform-ST are as follows:

Assumptions of Platform-ST	**Classification of assumptions**	**Mapping to Security Objectives of this Composite-ST**
A.Process-Sec-IC	IrPA (irrelevant)	n/a
A.Resp-Appl	CfPA (automatically fulfilled by the Composite-ST)	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL,

Assumptions of Platform-ST	Classification of assumptions	Mapping to Security Objectives of this Composite-ST
		O.GLOBAL_ARRAYS_CONFID, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.REALLOCATION O.GLOBAL_ARRAYS_INTEG
A.Key-Function	CfPA	O.SCP.IC, O.CIPHER, O.KEY-MNGT, O.PIN-MNGT, O.SCP.RECOVERY, O.SCP.SUPPORT

Table 15: Mapping of assumptions

The assumption A.Process-Sec-IC (security procedures to maintain confidentiality and integrity of the hardware TOE) only pertains to the hardware manufacturer.

A.Resp-Appl assumes that all user data owned by the embedded software are treated as defined for their specific application context. The PP demands that user data must be secured with respect to integrity and confidentiality (see [JCSPP], chapter 5 and 6), so all those objectives which cover the threats T.CONFID-APPLI-DATA and T.INTEG-APPLI-DATA fulfil the assumption. They have been entered into Table 15.

A.Key-Function assumes that key-dependent functions in the embedded software are not susceptible to leakage attacks. This is covered by the objective O.SCP.IC.

There is **no conflict** between **security environments** of this Composite-ST and the Platform-ST [IFX_STLite].

10.1.1.2 Security objectives

Security objectives see: chapter 6

This Composite-ST has the following security objectives which are directly related to the Platform-ST:

- O.SCP.IC
- O.SCP.RECOVERY
- O.SCP.SUPPORT

These objectives will be mapped to the following Platform-ST [IFX_STLite], chapter 5.1) objectives:

- O.Phys-Manipulation
- O.Phys-Probing
- O.Malfunction
- O.Leak-Inherent
- O.Leak-Forced

- O.Abuse-Func
- O.RND
- O.Add-Functions
- O.Mem-Access
- O.Cap_Avail_Loader

The mapping is shown below in table 16.

		Platform-ST												
		O.Leak-Inherent	O.Add-Functions	O.Mem-Access	O.RND	O.Phys-Probing	O.Malfunction	O.Phys-Manipulation	O.Leak-Forced	O.Abuse-Func	O.Cap_Avail_Loader	O.TDES	O.AES	O.SHA
Composite-ST	Obejctives for TOE_IC													
	O.SCP.RECOVERY						X							
	O.SCP.SUPPORT	X	X	X	X	X	X	X	X	X	X	X	X	X
	O.SCP.IC	X	X	X		X	X	X	X	X	X	X	X	X

Table 16 Mapping of objectives

O.SCP.RECOVERY matches to O.Malfunction because this allows 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 supports the well-functioning of the TSFs of the TOE (avoiding they are bypassed or altered). O.RND particularly provides a required low-level-security cryptographic function to the Java Card System.

O.SCP.IC matches to all listed objectives of the Platform-ST (except O.RND) because they describe features against physical attacks.

The Objectives for the Operational Environment (see 6.2) 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 [IFX_STLite].

10.1.1.3 Security requirements

Security Functional Requirements see chapter 8.1

Platform SFR	Relevance **RP_SFR:** relevant Platform SFR being used by the Composite ST **or** **IP_SFR:** irrelevant Platform SFR not used by the Composite ST	Correspondence in Composite ST
FPT_FLS.1	RP_SFR	FPT_FLS.1, FPT_RCV.3
FRU_FLT.2	RP_SFR	FPT_RCV.3
FCS_COP.1/TDES	RP_SFR	FCS_COP.1.1/3DES, TDES coprocessor is used
FCS_COP.1/AES	RP_SFR	FCS_COP.1.1/AES, AES coprocessor is used
FCS_CKM.4/TDES	RP_SFR	FCS_CKM.4
FCS_CKM.4/AES	RP_SFR	FCS_CKM.4
FCS_COP.1/RSA	IP_SFR	Asymmetric crypto library of the IFX platform is not used by the Composite TOE
FCS_CKM.1/RSA	IP_SFR	Asymmetric crypto library of the IFX platform is not used by the Composite TOE
FCS_COP.1/ECDSA	IP_SFR	Asymmetric crypto library of the IFX platform is not used by the Composite TOE
FCS_COP.1/ECDH	IP_SFR	Asymmetric crypto library of the IFX platform is not used by the Composite TOE
FCS_CKM.1/EC	IP_SFR	Asymmetric crypto library of the IFX platform is not used by the Composite TOE
FCS_COP.1/SHA	IP_SFR	Optional SHA crypto library of the IFX platform is not used by the Composite TOE
FPT_PHP.3	RP_SFR	FPT_PHP.3

Platform SFR	Relevance **RP_SFR:** relevant Platform SFR being used by the Composite ST **or** **IP_SFR:** irrelevant Platform SFR not used by the Composite ST	Correspondence in Composite ST
FCS_RNG.1/TRNG	RP_SFR	FCS_RNG.1.1, PTG.2 is used as input for DRG.4
FAU_SAS.1	IP_SFR	Test process before TOE Delivery is not used by the composite SFRs
FDP_ACC.1	RP_SFR	FDP_ACC.2/ADEL, FDP_ACC.2/FIREWALL
FDP_ACF.1	RP_SFR	FDP_ACF.1/ADEL, FDP_ACF.1/FIREWALL
FMT_MSA.3	RP_SFR	FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.3/CM
FMT_MSA.1	RP_SFR	FMT_MSA.1/ADEL, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_MSA.1/CM
FMT_SMF.1	IP_SFR	The access for the configuration registers of the MMU is not used by the composite SFRs
FDP_SDI.1	RP_SFR	FDP_SDI.2., hierarchical to: FDP_SDI.1
FDP_SDI.2	RP_SFR	FDP_SDI.2
FDP_ITT.1	RP_SFR	FDP_IFC.1.1/JCVM
FDP_IFC.1	RP_SFR	FDP_IFC.1/JCVM, FDP_IFC.2/CM
FMT_LIM.1	IP_SFR	Internal test features of the IFX platform are not accessible by the Composite TOE
FMT_LIM.2	IP_SFR	Internal test features of the IFX platform are not accessible by the Composite TOE
FMT_LIM.1/Loader	RP_SFR	FDP_SDI.2, FDP_UIT.1/CM Data
FMT_LIM.2/Loader	RP_SFR	FDP_SDI.2, FDP_UIT.1/CM Data
FDP_SDC.1	RP_SFR	FPT_PHP.3.1
FPT_ITT.1	RP_SFR	FDP_ACF.1/FIREWALL
FPT_TST.2	RP_SFR	FPT_TST.1

Table 17 Mapping of Platform and Composite SFRs and Relevance

FPT_FLS.1 matches to FPT_FLS.1 and FPT_RCV.3 as the Composite-TOE preserves a secure state when the Platform operates out of normal operating conditions, while the Platform ensures the robustness and operates always in a secure state.

FCS_COP.1/TDES and FCS_COP.1/AES match FCS_COP.1 as the Platform provides cryptographic support through a symmetric coprocessor for the composite product.

FDP_SDI.2 and FDP_UIT.1/CM Data match FMT_LIM.1/Loader and FMT_LIM.2/Loader as the Composite-TOE prevents stored user data to be disclosed or manipulated by unauthorised user.

FPT_PHP.3.1 matches FDP_SDC.1 as the Composite-TOE ensures the confidentiality of the user data.

FDP_ACF.1/FIREWALL matches FPT_ITT.1 as the Composite-TOE protects internal TSF data transfer.

10.1.2 Assurance requirements

The Composite-ST requires EAL 5 augmented by: ALC_DVS.2 and AVA_VAN.5.

The Platform-ST requires EAL 6 augmented with ALC_FLR.1.

Therefore, the assurance requirements for the composite TOE are a subset of the assurance requirements of the hardware TOE.

10.2 Overall no contradictions found

Overall there is **no conflict** between **security requirements** of this Composite-ST and the Platform-ST [IFX_STLite].

11 References, Abbreviations and Glossary

11.1 References

[AIS20] Anwendungshinweise und Interpretationen zum Schema (AIS), AIS 20, Version 3, 15.05.2013, Funktionalitätsklassen und Evaluierungsmethodologie für deterministische Zufallszahlengeneratoren, Zertifizierungsstelle des BSI.

[AIS31] Anwendungshinweise und Interpretationen zum Schema (AIS), AIS 31, Funktionalitätsklassen und Evaluierungsmethodologie für physikalische Zufallszahlengeneratoren, Version 3, 15.05.2013

[ANSIX962] ANSI X9.62:2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)

[CC1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model. Version 3.1, Revision 4, September 2012. CCMB-2012-09-001.

[CC2] Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements. Version 3.1, Revision 4, September 2012. CCMB-2012-09-002.

[CC3] Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components. Version 3.1, Revision 4, September 2012. CCMB-2012-09-003.

[EN14890] CEN/TC 224, EN 14890-1 Application Interface for smart cards used as Secure Signature Creation Devices — Part 1: Basic services

[FIPS46-3] Federal Information Processing Standards PUB 46-3, Data Encryption Standard, reaffirmed 1999 October 25

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

[FIPS180-4] Federal Information Processing Standards Publication 180-4, Secure Hash Standard, March 2012

[FIPS 186-3] Federal Information Processing Standards Publication FIPS PUB 186-3 DIGITAL SIGNATURE STANDARD (DSS) (with Change Notice), U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, 2009 June

[FIPS 197] Federal Information Processing Standards Publication 197, Advanced Encryption Standard, November 26, 2001

[GP221] GlobalPlatform Card Specification Version 2.2.1, January 2011

[GP AM D] GlobalPlatform Card Specification Version 2.2 – Amendment D, Secure Channel Protocol 03, Version 1.1, September 2009

[GP ID_Config] GlobalPlatform Card ID Configuration Version 1.0, December 2011

[GP CIC] GlobalPlatform Card Common Implementation Configuration Version 1.0, February 2014

[GP SE_Config] GlobalPlatform Card Secure Element Configuration Version 1.0, October 2012

[IFX_Cert] Certification report BSI-DSZ-CC-0951-2015 for Infineon Security Controller M5073 G11 with optional RSA2048/4096 v2.03.008, EC v2.03.008,SHA-2 v1.01 and Toolbox v2.03.008 libraries and with specific IC dedicated software (firmware) from Infineon Technologies AG, 11.11.2015
Assurance Continuity Reassessment Report BSI-DSZ-CC-0951-2015-RA-01, Infineon Security Controller M5073 G11 with optional RSA2048/4096 v2.03.008, EC v2.03.008,SHA-2 v1.01 and Toolbox v2.03.008 libraries and with specific IC dedicated software (firmware) from ,Infineon Technologies AG, 31.05.2017

[IFX_STLite] Security Target Lite for BSI-DSZ-CC-0951-2015, Version 1.2, 2017-05-10, Infineon

[ISO9796-2] ISO/IEC 9796-2, Information Technology – Security Techniques – Digital Signature Schemes giving message recovery – Part 2: Integer factorisation based mechanisms, 2002

[ISO9797-1] ISO/IEC 9797-1:2011: Information technology – Security techniques –Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher.

[ISO15946] ISO/IEC 15946-5:2009, Cryptographic techniques based on elliptic curves

[JCAPI22] Java Card 2.2.2 Application Programming Interface. March 2006. Published by Sun Microsystems, Inc.

[JCAPI3] Java Card Platform, versions 3.0 (March 2008) and 3.0.1 (April 2009), Classic Edition, Application Programming Interface, March 2008. Published by Sun Microsystems, Inc.

[JCAPI304] Java Card API, Classic Edition. Version 3.0.4., 2011 (Oracle)

[JCRE22] Java Card 2.2.2 Runtime Environment (JCRE) Specification. March 2006. Published by Sun Microsystems, Inc.

[JCRE304] Java Card 3 Platform Runtime Environment Specification, Classic Edition. Version 3.0.4., September 2011 (Oracle)

[JCSPP] Java Card Protection Profile - Open Configuration, May 2012, Version 3.0, Oracle Corporation

[JCVM22] Java Card 2.2.2 Virtual Machine (JCVM) Specification. March 2006. Published by Sun Microsystems, Inc.

[JCVM304] Java Card 3 Platform Virtual Machine Specification, Classic Edition. Version 3.0.4., September 2011 (Oracle)

[JIL] Certification of "open" smart card products, Version 1.1 (for trial use), 4 February 2013, Joint Interpretation Library

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

[PKCS1] PKCS #1: RSA Encryption Standard – An RSA Laboratories Technical Note, Version 2.1, February, 2003

[PKCS3] PKCS#3: Diffie-Hellman Key Agreement Standard, An RSA Laboratories Technical Note, Version 1.4, Revised November 1, 1993

[PKCS5] PKCS #5: Password-Based Encryption Standard, Version 2.1.

[PP0035] Security IC Platform Protection Profile, Version 1.0, 15 June 2007, BSI-CC-PP-0035-2007

[PP0084] Security IC Platform Protection Profile with Augmentation Packages Version 1.0, 19.02.2014, BSI-CC-PP-0084-2014

[PP_JCS] Java Card Protection Profile Collection, Version 1.1, May 2006.

[RFC2437] PKCS #1: RSA Cryptography Specifications, Version 2.0, October 1998

[RFC4493] RFC 4493, The AES-CMAC Algorithm, June 2006

[RFC5639] RFC 5639, Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, BSI and secunet, March 2010, ISSN 2070-1721

[RSA] RFC 3447, J. Jonsson, B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", February 2003

[RSASSA-PSS] RFC 4056, J.Schaad, "Use of the RSASSA-PSS Signature Algorithm in Cryptographic Message Syntax (CMS)", June 2005

[RSA-SHA-RFC2409] RFC 2409, D. Harkins, D. Carrel, "The Internet Key Exchange (IKE), November 1998

[SEC2] Standards for efficient cryptography, SEC 2: Recommended Elliptic Curve Domain Parameters, Certicom Research, September 20, 2000, Version 1.0

[SP800-38a] Recommendation for Block Cipher Modes of Operation, NIST Special Publication 800-38A, December 2001

[SP800-38b] Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, NIST Special Publication 800-38B, May 2005

[SP800-67] National Institute of Standards and Technology, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Special Publication 800-67, Revised January 2012

[SP800-90A] NIST Special Publication 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators, January 2012

[SUPP] Supporting Document, Mandatory Technical Document, Composite product evaluation for Smart Cards and similar devices, April 2012, Version 1.2, CCDB-2012-04-001.

[TR-3111] Technical Guideline TR-03111 Elliptic Curve Cryptography, TR-03111, version 2.0, 28.06.2012, Bundesamt für Sicherheit in der Informationstechnik (BSI)

[UGPre] Preparative Guidance Sm@rtCafé® Expert 7.0 C3, Giesecke & Devrient, Version 3.6

[UGOpe] Operative Guidance Sm@rtCafé® Expert 7.0 C3, Giesecke & Devrient, Version 5.2

11.2 Abbreviations

API Application Programming Interface

CAD Card Acceptance Device

CAP Converted Applet

CC Common Criteria

DAP Data Authentication pattern

DES Data Encryption Standard

DS Dedicated software

EAL Evaluation Assurance Level

ECC Elliptic Curve Cryptography

HW Hardware

IC Integrated Circuit

IP-SFR Irrelevant Platform Security Functional Requirement

IT Information Technology

JCRE Java Card Runtime Environment

JCS Java Card System

JCVM Java Card Virtual Machine

OS Operating System

PCD Proximity Coupling Device

PIN Personal Identification Number

PP Protection Profile

RAM Random Access memory

ROM Read-Only Memory

RP-SFR Relevant Platform Security Functional Requirement

RSA Rivest, Shamir and Adleman

SCP Smart Card Platform

SF Security Function

SFP Security Function Policy

SFR Security Functional Requirement

SHA Secure Hash Algorithm

SIO Serial Input Output

ST Security Target

SW Software

TOE Target of Evaluation

TSC TSF Scope of Control

TSF TOE Security Functions

TSFI TSF Interface

TSP TOE Security Policy

11.3 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 package*s. *AIDs* are administered by the International Standards Organization (ISO), so they can be used as unique identifiers. *AIDs* are also used in the security policies (see "*Context*" below): applets' *AIDs* are related to the selection mechanisms, *package*s' *AIDs* are used in the enforcement of the *firewall*. **Note:** although they serve different purposes, they share the same name space.
APDU	Application Protocol Data Unit, an ISO 7816-4 defined communication format between the card and the off-card applications. Cards receive requests for service from the CAD in the form of *APDU*s. These are encapsulated in Java Card System by the **`javacard.framework.APDU`** *class* ([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 manager	Application with specific rights which is responsible for the administration of the Java smart card.
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).
DAP	Data Authentication pattern are used to authenticate the origin and/or integrity of the data through Hash or MAC or other cryptographic methods.
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*).
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.
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.
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.

open configuration	Configuration of a Java smart card which allows post-issuance loading of applets.
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.
PCD	Proximity Coupling Device. The PCD is a contactless card reader device.
PICC	Proximity Card. The PICC is a card with contactless capabilities.
RAM	Random Access Memory, is a type of computer memory that can be accessed randomly.
SCP	Smart Card Platform. It is comprised of the integrated circuit, the 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