Ucard UBJ31-G11 V1.1 Security Target Lite V1.0

Version	V1.0
Writer	UBIVELOX
Date	2012-09-20

History

Document Revision	Modification	Approval date	Writer
V1.0	Final confirmation	2012-09-20	UBIVELOX

Table of Contents

1 ST Introduction

1.1 ST reference

Title	Ucard UBJ31-G11 V1.1 Security Target Lite V1.0
Version	Version 1.0
Date	2012.09.20
Author(s)	UBIVELOX
CC level	EAL4+ (ALC_DVS.2, AVA_VAN.5)
Key word	Smart card, COS, IC, Java card, Global Platform

Table1. ST reference

1.2 TOE reference

name	Ucard UBJ31-G11 V1.1
version	Version 1.1
chip identifier	SB23YR80B
chip certificate reference	ANSSI-CC-2010/02

Table2. TOE reference

1.3 TOE overview

The TOE Type is a smartcard(Java Card Platform), which is composed of operative system Embedded software and IC SB23YR80B.

The TOE is compliant with Java Card 2.2.2(Java Card 2.2.2 Runtime Environment Specification [JCRE], Java Card 2.2.2 Virtual Machine Specification [JCVM], Java Card 2.2.2 Application Programming Interfaces [JCAPI]) and Visa GlobalPlatform 2.1.1-configuration 3 standards.

It provides the security level of EAL4+ augmented with ALC_DVS.2 and AVA_VAN.5 and allows loading and deleting applications, which are developed by the customers. Thus, it allows for multiple applications to run on a single TOE and provides security features to ensure secure interoperability of applications.

The TOE processes personal information in secure manner. By using this secure information, the TOE can improve the reliability.
The TOE provides a wide area of physical protection measures and implements the logical security measures; refer to 1.4.2 for TOE security functionality.
By using the TOE, the final user can store private key, personal certificates, personal information or buy something or use transport system in safety.

The examples of TOE intended usage are:
• Financial applications, like Credit/Debit ones, stored value purse, or electronic commerce, among others.
• Transport and ticketing, granting pre-paid access to a transport system like the metro and bus lines of a city.

• Telephony, through the subscriber identification module (SIM) for digital mobile telephones.
• Personal identification, for granting access to secured sites or providing identification credentials to participants of an event.
• Electronic passports and identity cards.
• Secure information storage, like health records, or health insurance cards.
• Loyalty programs, like the “Frequent Flyer” points awarded by airlines. Points are added and deleted from the card memory in accordance with program rules. The total value of these points may be quite high and they must be protected against improper alteration in the same way that currency value is protected.

1.4 TOE description

1.4.1 TOE Architecture

The TOE consists of:
• Smart card platform (SCP) (parts of the IC SB23YR80B and Operation System)
• Embedded software (Java Card Virtual Machine (JCVM), Java Card Runtime Environment (JCRE), Java Card API (JCAPI), Card Manager & GP API)

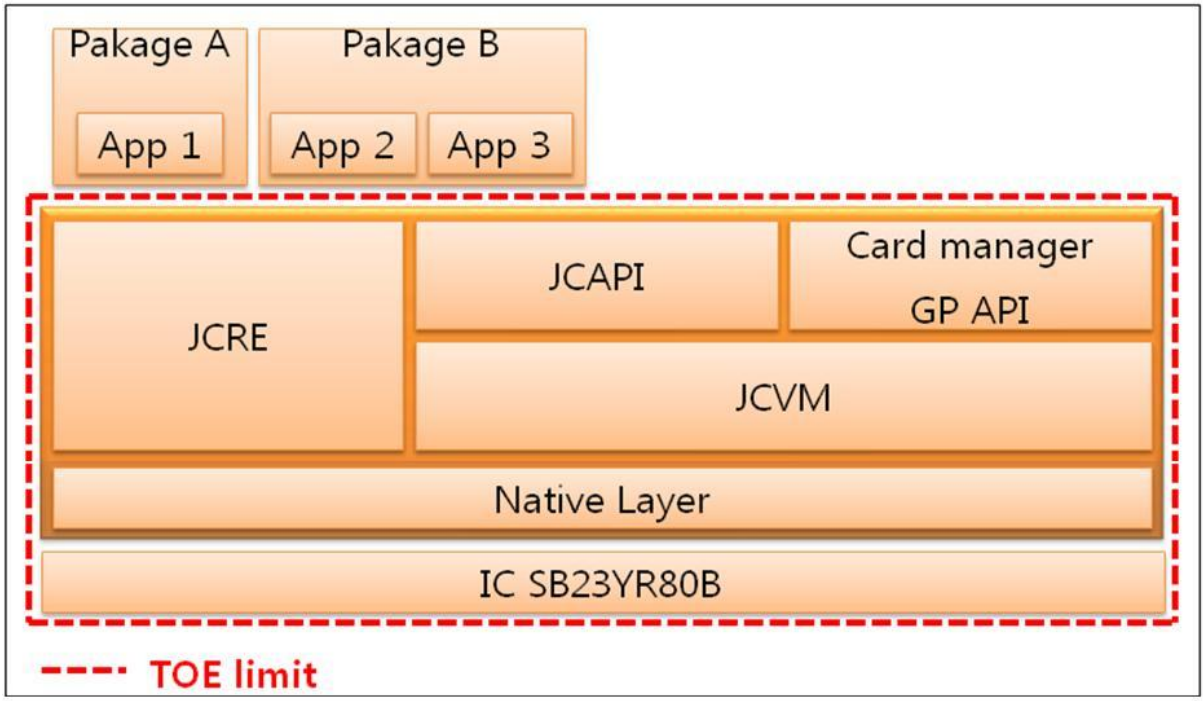


Figure1. TOE Architecture

The IC provides:
- maintain the integrity and the confidentiality of the content of the Security IC memories as required by the context of the Security IC Embedded Software and
- maintain the correct execution of the Security IC embedded Software.

The Native Layer includes operating system and NESCRTPT library (NesLib). It provides the basic functionalities (memory management, I/O management and

cryptographic libraries) with native interface with the dedicated IC. The cryptographic library provides high-level routines to perform RSA, SHA, AES and ECC operation using NESCRYPT for highly secure IC.

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

The basic runtime security feature imposed by the JCRE enforces isolation of applets using an applet firewall. It prevents objects created by one applet from being used by another applet without explicit sharing. This prevents unauthorized access to the fields and methods of class instances, as well as the length and contents of arrays.
The applet firewall is considered as the most important security feature. It enables complete isolation between applets or controlled communication through additional mechanisms that allow them to share objects when needed. The JCRE allows such sharing using the concept of “shareable interface objects” (SIO) and static public variables. The JCVM should ensure that the only way for applets to access any resources are either through the JCRE or through the Java Card API (or other vendor-specific APIs). This objective can only be guaranteed if applets are correctly typed.
The Card Manager is conformant to the VISA Global Platform Card Specification 2.1.1 and is responsible for the management of applets in the card.

The delivered TOE component is as below.

Classification (Delivery Type)		**TOE & TOE component**	**Name**
TOE	SW	ROM code (ROM file + EEPROM file)	UBJ31-G11_DEL
	HW	Chip Identifier	SB23YR80B
Guidance Documents (File)		Operational user guidance	Operational User Guidance
		Preparative procedures	Preparative procedures

Table3. TOE Component

1.4.2 TOE security functionality

The TOE complies with four major industry standards:
• Sun’s Java Card 2.2.2, which consists of the Java Card 2.2.2 Virtual Machine (JCVM) [JCVM222], Java Card 2.2.2 Runtime Environment (JCRE) and the Java Card 2.2.2 [JCRE222], Application Programming Interface (JCAPI). [JCAPI222]
• The Global Platform Card Specification version 2.1.1 [GP]
- Delegated Management is not support and APDU command conforms to Visa GlobalPlatform 2.1.1 Card Implementation Requirements v2.0.
• Visa GlobalPlatform 2.1.1 Card Implementation Requirements v2.0 [VGP]
- Implemented by Configuration 3 (Multiple Security Domains and DAP Verification)
• Finance IC card standard revision - Open platform – October 2010 [FICCS]

According to the [FICCS], only Korean Package for cryptography algorithm SEED has implemented on the TOE.

The TOE provides a wide area of physical protection measures and implements the logical security measures, being them:

• Cryptographic algorithms and functionality:
  a. DES (56 bit keys) for en/decryption (CBC and ECB)
  b. TDES (112, 168 bit keys) for en/decryption (CBC and ECB)
  c. AES (Advanced Encryption Standard) with key length of 128, 192 and 256 bit for en/decryption (CBC and ECB)
  d. RSA (512 up to 2048 bits keys) for key generation
  e. RSA (512 up to 2048 bits keys) en/decryption and signature generation and verification
  f. RSA CRT (512 up to 2048 bits keys) for key generation
  g. RSA CRT (512 up to 2048 bits keys) en/decryption and signature generation and verification
  h. Hash Algorithm - SHA-1, SHA-256
  i. ECDSA (112 up to 521 bits keys) for signature generation and verification
  j. SEED (128 bit keys) for en/decryption
  k. Random number generation according to class P2 of AIS-31
  l. CRC (16 bit)
• Java Card 2.2.2 functionality:
  a. Remote Method Invocation
  Supports the remote methods that can be invoked remotely from CAD.
  b. Multiple Logical Channel.
  Supports multiple logical channels which allow a terminal to open up to eight channels (contact 4 channels and contactless 4 channels) with the smart card, one session per logical channel. (Logical channels functionality is described in detail in [GP] & [VGP])
  c. Garbage Collector
  Reclaims deallocated data automatically during the execution of a program
  d. Firewall
  The mechanism in the Java Card technology for ensuring applet isolation and object sharing. The firewall prevents an applet in one context from unauthorized access to objects owned by the Java Card RE or by an applet in another context.

• Global Platform 2.1.1 functionality:
  a. Issuer Security Domain.
  Operates as the mandatory on-card representative of the Card Issuer which has capability of loading, installing, and deleting application that belong either to the Card Issuer or to other Application Provider.
  b. Supplementary Security Domain.
  Operates as the on-card representative of an Application Provider or Controlling Authority.
  c. Public key DAP Verification.
  Supports verification of application code integrity and authenticity before the application code is loaded, installed and made available to the Cardholder on behalf of an Application Provider.
  d. Mandated DAP Verification.

Supports verification of application code integrity and authenticity before the application code is loaded, installed and made available to the Cardholder on behalf of a Controlling Authority.

e. Secure Channel Protocol 02.

Provides a secure communication channel between a card and an off-card entity during an Application Session.

f. CVM interface supporting Global PIN.

Provides support for CVM management which is responsible for Cardholder verification, including velocity checking.

• General Functionalities:

a. Communication protocols:
   - ISO 7816 T=0
   - ISO 7816 T=1
   - ISO 14443 T=CL Type B (contact-less)

b. Support various baud rates for Communication Protocols.

c. Protection against Physical Probing and against malfunctions.

d. Security data integrity.

e. Security alarms in case of detect a security violation.

f. Atomicity of critical operations.

1.4.3 TOE environment

The TOE requires a CAD device and an application that implements communication with the card in order to operate in a correct way. These are not included under the TOE scope.

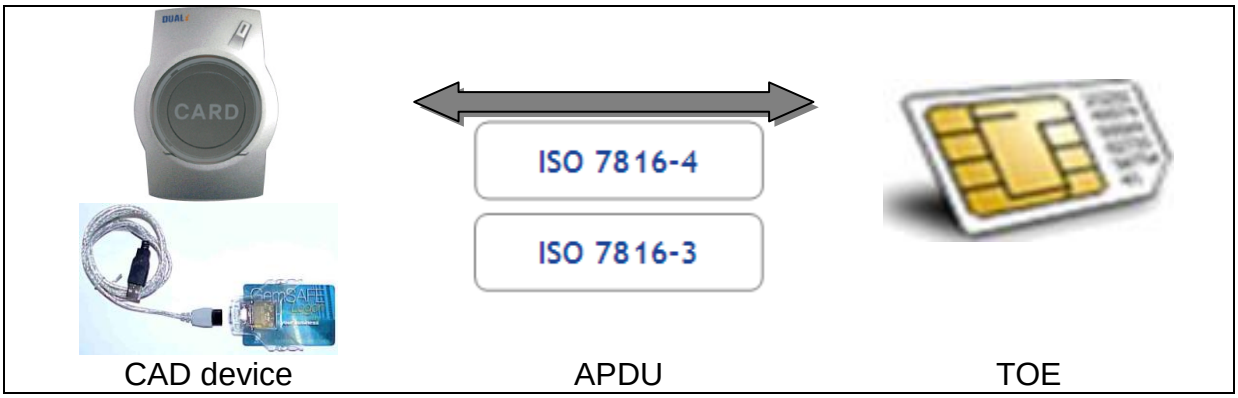


Figure2. TOE environment

Applications installed on a TOE can be selected for execution when the card is inserted into a CAD device.

The examples of TOE intended usage are:

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

• Transport and ticketing, granting pre-paid access to a transport system like the metro and bus lines of a city.

• Telephony, through the subscriber identification module (SIM) for digital mobile telephones.

• Personal identification, for granting access to secured sites or providing

identification credentials to participants of an event.
• Electronic passports and identity cards.
• Secure information storage, like health records, or health insurance cards.
• Loyalty programs, like the “Frequent Flyer” points awarded by airlines. Points are added and deleted from the card memory in accordance with program rules. The total value of these points may be quite high and they must be protected against improper alteration in the same way that currency value is protected.

1.4.4 TOE Life Cycle

The TOE life cycle is part of the product life cycle, i.e. the Java Card platform with applications, which goes from product development to its usage by the final user.

Phase 1	Security IC Embedded Software Development	The **IC Embedded Software Developer** is in charge of • smartcard embedded software development including the development of Java applets • specification of IC pre-personalization requirements.
Phase 2	Security IC Development	The **IC Developer** • designs the IC, • develops IC Dedicated Software, • provides information, software or tools to the IC Embedded Software Developer, and • receives the smartcard embedded software from the developer, through trusted delivery and verification procedures. From the IC design, IC Dedicated Software and Smartcard Embedded Software, the IC Developer • constructs the smartcard IC database, necessary for the IC photomask fabrication.
Phase 3	Security IC Manufacturing	The **IC Manufacturer** is responsible for • producing the IC through three main steps: IC manufacturing, IC testing, and IC pre-personalization The IC Mask Manufacturer • generates the masks for the IC manufacturing based upon an output from the smartcard IC database
Phase 4	Security IC packaging	The **IC Packaging Manufacturer** is responsible for • IC packaging and testing.
Phase 5	Composite Product Integration	The **Composite Product Manufacturer** is responsible for •smart card product finishing process and testing.
Phase 6	Personalization	The **Personaliser** is responsible for • smart card (including applet) personalization and final tests. Other applets may be loaded onto the chip at the personalization process.
Phase 7	Operational Usage	The **Consumer of Composite Product** is responsible for • smartcard product delivery to the smartcard end-user, and the end of life process.

Table4. TOE Life Cycle

The evaluation process is limited to phases 1 to 3.
The delivery of the JCS is either in phase 3.
The delivery of the smart card product is in phase 7.

1.4.5 The user and roles

The users of the TOE include the following people and institutions

IC Embedded Software Developer

The IC Embedded Software Developer is the organization responsible for designing and implementing the software masked on the IC. This includes the following components of the TOE: Card Manager, Java Card Runtime Environment, and Operating System.

IC Developer

The IC Developer designs the chip and its Dedicated Software (DS).

Manufacturers

The IC Manufacturer integrates the Embedded Software within the IC. This is usually known as the "masking" process.

The IC Packaging Manufacturer integrates the masked IC with the carrier (a plastic card, a passport booklet, etc) in accordance with the Card Issuer's requirements.

The Composite Product Manufacturer is store, pre-personalize the JCS and potentially conduct tests on behalf of the IC Embedded Software Developer.

Personalizer

The Smart Card Personalizer personalizes the card by loading the cardholder data as well as cryptographic keys and PINs. For this TOE, the personalizer is the Card Issuer.

Card Administrator

The person or organization that has ultimate control of the card, within the policy constraints set by the Card Issuer, with regard to card content and card Life Cycle management.

End-user, Signatory, Card Holder

The Signatory is the End-user in the usage phase (phase 7) and owns the TOE. The card is personalized with his or her identification and secrets.

1.5 ST organization

Chapter 1 provides narrative descriptions of the TOE.
Chapter 2 shows the ST claims conformance to the CC, PP and package.
Chapter 3 describes the main security issues of the Java Card System and its environment addressed in the [JCSPP] with which ST claims conformance.
Chapter 4 describes threats, organizational security policies, and assumptions on the TOE and/or TOE operational environment.
Chapter 5 describes how the solution to the security problem is divided between objectives for the TOE and security objectives for the operational environment of the TOE.
Chapter 6 describes new components those not included in [CCPART2] or [CCPART3].
Chapter 7 describes security requirements where a translation of the security objectives for the TOE into SFRs. Additionally, this chapter defines the SARs.
Chapter 8 describes a TOE summary specification which shows how the SFRs are

implemented in the TOE.
Chapter 9 provides references to other documents, terms and definitions, and abbreviated terms.

1.6 Conventions

The notation, formatting and conventions used in this ST are consistent with the [JCSPP].
The CC allows several operations to be performed on functional requirements; assignment, iteration, refinement and selection. Each of these operations is used in this ST, and they are also consistent with the [JCSPP].
Iteration: The ST author performs an iteration operation by including multiple requirements based on the same component. The result of iteration is marked by '/' with unique name following the component identifier, e.g., FDP_ACC.2/FIREWALL.
Assignment: An assignment operation occurs where a given component contains an element with a parameter that may be set by the ST author. It is used to assign specific values to unspecified parameters (e.g. password length). The result of assignment is shown in **bold** text.
Selection: The selection operation occurs where a given component contains an element where a choice from several items has to be made by the ST author. The result of selection is shown in **bold** text.
Refinement: The ST author performs a refinement by altering that requirement by fulfilling refinement rules specified in the CC. The result of refinement is shown in **bold** text with 'refinement' notification.

Also, "Application Notes" are provided to help to clarify the intent of a requirement, identify implementation choices or to define "Pass/Fail" criteria for a requirement. Application Notes will follow relevant requirements where appropriate. There are two types of application note, i.e. from [JCSPP] and [ST].

2 Conformance claims

2.1 CC conformance claim

The ST claims the conformance to the following version of the CC:

- Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, July 2009, Version 3.1 Revision 3 Final, CCMB-2009-07-001 [CCPART1]
- Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements, July 2009, Version 3.1 Revision 3 Final, CCMB-2009-07-002 [CCPART2]
- Common Criteria for Information Technology Security Evaluation, Part 2: Security assurance requirements, July 2009, Version 3.1 Revision 3 Final, CCMB-2009-07-003 [CCPART3]

Conformance to the CC is claimed as follows:

- Part 2: extended
- Part 3: conformant

2.2 PP claim

The ST claims conformance to the Java Card™ System Protection Profile Open Configuration, Version 2.6, April 19, 2010 [JCSPP].
The TOE is composite TOE based on the certified IC chip. The IC chip has been certified separately according to [ICST] claiming [PP0035].

2.3 Package claim

The ST claims conformance to EAL4+ augmented with ALC_DVS.2 and AVA_VAN.5 as stated in [JCSPP].

2.4 Conformance rationale

2.4.1 Conformance rationale for the TOE type

The TOE type of [JCSPP] is Java Card System (JCRE, JCVM and JCAPI) along with the additional native code embedded in a Smart Card Platform. And [JCSPP] requires that the TOE type of the ST that declares conformity to [JCSPP] is the Smart Card Platform (IC and OS) along with the native application (if any), pre-issuance applets (if any) and Java Card System.
The TOE type of this ST is the Smart Card Platform along with Java Card System and Card Manager. There is no native application or pre-issuance applets on the TOE.
Therefore, the TOE type is consistent with the TOE type in [JCSPP].

2.4.2 Conformance rationale for the security problem definition

The following table shows the security problem definition in this ST is consistent with those of [JCSPP].

[JCSPP]	ST	Rationale
T.CONFID-APPLI-DATA	T.CONFID-APPLI-DATA	Same
T.CONFID-JCS-CODE	T.CONFID-JCS-CODE	Same
T.CONFID-JCS-DATA	T.CONFID-JCS-DATA	Same
T.INTEG-APPLI-CODE	T.INTEG-APPLI-CODE	Same
T.INTEG-APPLI-CODE.LOAD	T.INTEG-APPLI-CODE.LOAD	Same
T.INTEG-APPLI-DATA	T.INTEG-APPLI-DATA	Same
T.INTEG-APPLI-DATA.LOAD	T.INTEG-APPLI-DATA.LOAD	Same
T.INTEG-JCS-CODE	T.INTEG-JCS-CODE	Same
T.INTEG-JCS-DATA	T.INTEG-JCS-DATA	Same
T.SID.1	T.SID.1	Same
T.SID.2	T.SID.2	Same
T.EXE-CODE.1	T.EXE-CODE.1	Same
T.EXE-CODE.2	T.EXE-CODE.2	Same
T.EXE-CODE-REMOTE	T.EXE-CODE-REMOTE	Same
T.NATIVE	T.NATIVE	Same
T.RESOURCES	T.RESOURCES	Same
T.DELETION	T.DELETION	Same
T.INSTALL	T.INSTALL	Same
T.OBJ-DELETION	T.OBJ-DELETION	Same
T.PHYSICAL	T.PHYSICAL	Same
-	T.ACCESS	Due to the scope of the TOE in the ST, this threat is additionally defined for the Smart Card Platform and GP.
-	T.OS_OPERATE	Due to the scope of the TOE in the ST, this threat is additionally defined for the Smart Card Platform and GP.
-	T.LEAKAGE	Due to the scope of the TOE in the ST, this threat is additionally defined for the Smart Card Platform and GP.
-	T.FAULT	Due to the scope of the TOE in the ST, this threat is additionally defined for the Smart Card Platform and GP.
-	T.RND	Due to the scope of the TOE in the ST, this threat is additionally defined for the

[JCSPP]	ST	Rationale
		Smart Card Platform and GP.
OSP.VERIFICATION	OSP.VERIFICATION	Same
A.DELETION	OSP.DELETION	Originally this organizational security policy was addressed under an assumption A.DELETION in the [JCSPP]. The TOE includes the Card Manager to delete applets securely, thus the ST author moved the related assumption to the section of the security organizational policy to address this security aspects properly.
-	OSP.ROLES	Due to the scope of the TOE in the ST, this OSP is additionally defined for the Smart Card Platform and GP.
-	OSP.INITIAL_LIFE CYCLE_STATES	Due to the scope of the TOE in the ST, this OSP is additionally defined for the Smart Card Platform and GP.
-	OSP.CARD_ADMI NISTRATOR_PRE-APPROVAL	Due to the scope of the TOE in the ST, this OSP is additionally defined for the Smart Card Platform and GP.
-	OSP.APPLICATIO N_PROVIDER_PR E-APPROVAL	Due to the scope of the TOE in the ST, this OSP is additionally defined for the Smart Card Platform and GP.
-	OSP.LOAD_FILE_ VERIFICATION	Due to the scope of the TOE in the ST, this OSP is additionally defined for the Smart Card Platform and GP.
-	OSP.APPLICATIO N_CODE_VERIFIC ATION	Due to the scope of the TOE in the ST, this OSP is additionally defined for the Smart Card Platform and GP.
-	OSP.SECURE_CO MMUNICATION	Due to the scope of the TOE in the ST, this OSP is additionally defined for the Smart Card Platform and GP.
-	OSP.CARDHOLDE R_VERIFICATION	Due to the scope of the TOE in the ST, this OSP is additionally defined for the Smart Card Platform and GP.
-	OSP.PROD_PROC ESS	Due to the scope of the TOE in the ST, this OSP is additionally defined for the Smart Card Platform and GP.
-	OSP.CRYPTO	Due to the scope of the TOE in the ST, this OSP is additionally defined for the Smart Card Platform and GP.
A.APPLET	A.APPLET	Same
A.VERIFICATION	A.VERIFICATION	Same
-	A.KEY_MANAGEM ENT	Due to the scope of the TOE in the ST, this assumption is additionally defined for the Smart Card Platform and GP.

[JCSPP]	ST	Rationale
-	A.CVM	Due to the scope of the TOE in the ST, this assumption is additionally defined for the Smart Card Platform and GP.
-	A.ACTORS	Due to the scope of the TOE in the ST, this assumption is additionally defined for the Smart Card Platform and GP.

2.4.3 Conformance rationale for security objectives

The following table shows the security objectives in this ST is consistent with those of [JCSPP].

[JCSPP]	ST	Rationale
O.SID	O.SID	Same
O.FIREWALL	O.FIREWALL	Same
O.GLOBAL_ARRAYS_CONFID	O.GLOBAL_ARRAYS_CONFID	Same
O.GLOBAL_ARRAYS_INTEG	O.GLOBAL_ARRAYS_INTEG	Same
O.NATIVE	O.NATIVE	Same
O.OPERATE	O.OPERATE	Same
O.REALLOCATION	O.REALLOCATION	Same
O.RESOURCES	O.RESOURCES	Same
O.ALARM	O.ALARM	Same
O.CIPHER	O.CIPHER	Same
O.KEY-MNGT	O.KEY-MNGT	Same
O.PIN-MNGT	O.PIN-MNGT	Same
O.REMOTE	O.REMOTE	Same
O.TRANSACTION	O.TRANSACTION	Same
O.OBJ-DELETION	O.OBJ-DELETION	Same
O.DELETION	O.DELETION	Same
O.LOAD	O.LOAD	Same
O.INSTALL	O.INSTALL	Same
OE.CARD-MANAGEMENT	O.CARD-MANAGEMENT	Originally this security objective was addressed under OE.CARD-MANAGEMENT in the [JCSPP]. The TOE includes the Card Manager to manage applets securely, thus the ST author reassigned the related security objective for the operational environment to the security objective for the TOE to address this security aspects properly.
OE.SCP.IC	O.SCP.IC	Originally this security objective was addressed under OE.SCP.IC in the [JCSPP]. The TOE includes the SCP, especially the certified IC chip, and TOE

[JCSPP]	ST	Rationale
		is a subject to the composite evaluation against the [COMP-EVAL], thus the ST author reassigned the related security objective for the operational environment to the security objective for the TOE to address this security aspects properly.
OE.SCP.RECOVERY	O.SCP.RECOVERY	Originally this security objective was addressed under OE.SCP.RECOVERY in the [JCSPP]. The TOE includes the SCP, especially the certified IC chip, and TOE is a subject to the composite evaluation against the [COMP-EVAL], thus the ST author reassigned the related security objective for the operational environment to the security objective for the TOE to address this security aspects properly.
OE.SCP.SUPPORT	O.SCP.SUPPORT	Originally this security objective was addressed under OE.SCP.SUPPORT in the [JCSPP]. The TOE includes the SCP, especially the certified IC chip, and TOE is a subject to the composite evaluation against the [COMP-EVAL], thus the ST author reassigned the related security objective for the operational environment to the security objective for the TOE to address this security aspects properly.
-	O.PROTECT_DATA	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	O.OS_OPERATE	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	O.SIDE_CHANNEL	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	O.FAULT_PROTECT	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	O.RND	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.

[JCSPP]	ST	Rationale
-	O.ROLES	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	O.CARD_ADMIN	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	O.APPLICATION_PROVIDER_PRE-APPROVAL	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	O.LOAD_FILE_VERIFICATION	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	O.APPLICATION_CODE_VERIFICATION	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	O.SECURE_COMM	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	O.CARDHOLDER_VERIFICATION	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
OE.APPLET	OE.APPLET	Same
OE.VERIFICATION	OE.VERIFICATION	Same
-	OE.ACTORS	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	OE.INITIAL_LIFECYCLE_STATES	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	OE.PROD_PROCESS	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	OE.KEY_MANAGEMENT	Due to the scope of the TOE in the ST, this security objective is additionally defined for the Smart Card Platform and GP.
-	OE.CVM	Due to the scope of the TOE in the ST,

[JCSPP]	ST	Rationale
		this security objective is additionally defined for the Smart Card Platform and GP.

2.4.4 Conformance rationale for security requirements

The following table shows the security requirements in this ST is consistent with those of [JCSPP].

- CoreG_LC

[JCSPP]	ST	Rationale
FDP_ACC.2.1/FIREWALL	FDP_ACC.2.1/FIREWALL	Same
FDP_ACC.2.2/FIREWALL	FDP_ACC.2.2/FIREWALL	Same
FDP_ACF.1.1/FIREWALL	FDP_ACF.1.1/FIREWALL	Same
FDP_ACF.1.2/FIREWALL	FDP_ACF.1.2/FIREWALL	Same
FDP_ACF.1.3/FIREWALL	FDP_ACF.1.3/FIREWALL	Same
FDP_ACF.1.4/FIREWALL	FDP_ACF.1.4/FIREWALL	Same
FDP_IFC.1.1/JCVM	FDP_IFC.1.1/JCVM	Same
FDP_IFF.1.1/JCVM	FDP_IFF.1.1/JCVM	Same
FDP_IFF.1.2/JCVM	FDP_IFF.1.2/JCVM	Same
FDP_IFF.1.3/JCVM	FDP_IFF.1.3/JCVM	Assignment operation completed. “equivalent”
FDP_IFF.1.4/JCVM	FDP_IFF.1.4/JCVM	Assignment operation completed. “equivalent”
FDP_IFF.1.5/JCVM	FDP_IFF.1.5/JCVM	Assignment operation completed. “equivalent”
FDP_RIP.1.1/OBJECTS	FDP_RIP.1.1/OBJECTS	Same
FMT_MSA.1.1/JCRE	FMT_MSA.1.1/JCRE	Same
FMT_MSA.1.1/JCVM	FMT_MSA.1.1/JCVM	Same
FMT_MSA.2.1/FIREWALL_JCVM	FMT_MSA.2.1/FIREWALL_JCVM	Same
FMT_MSA.3.1/FIREWALL	FMT_MSA.3.1/FIREWALL	Same
FMT_MSA.3.2/FIREWALL	FMT_MSA.3.2/FIREWALL	Same
FMT_MSA.3.1/JCV	FMT_MSA.3.1/JCV	Same

[JCSPP]	ST	Rationale
M	M	
FMT_MSA.3.2/JCVM	FMT_MSA.3.2/JCVM	Same
FMT_SMF.1.1	FMT_SMF.1.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Requirement itself is all the same.
FMT_SMR.1.1	FMT_SMR.1.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Requirement itself is all the same.
FMT_SMR.1.2	FMT_SMR.1.2/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Requirement itself is all the same.
FCS_CKM.1.1	FCS_CKM.1.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Assignment operation completed. “more restrictive”
FCS_CKM.2.1	FCS_CKM.2.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Assignment operation completed. “more restrictive”
FCS_CKM.3.1	FCS_CKM.3.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Assignment operation completed. “more restrictive”
FCS_CKM.4.1	FCS_CKM.4.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Assignment operation completed. “more restrictive”
FCS_COP.1.1	FCS_COP.1.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Assignment operation completed. “more restrictive”
FDP_RIP.1.1/ABORT	FDP_RIP.1.1/ABORT	Same
FDP_RIP.1.1/APDU	FDP_RIP.1.1/APDU	Same
FDP_RIP.1.1/bArray	FDP_RIP.1.1/bArray	Same
FDP_RIP.1.1/KEY	FDP_RIP.1.1/KEY	Same

[JCSPP]	ST	Rationale
S	S	
FDP_RIP.1.1/TRANSIENT	FDP_RIP.1.1/TRANSIENT	Same
FDP_ROL.1.1/FIREWALL	FDP_ROL.1.1/FIREWALL	Same
FDP_ROL.1.2/FIREWALL	FDP_ROL.1.2/FIREWALL	Same
FAU_ARP.1.1	FAU_ARP.1.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Assignment operation completed. “more restrictive”
FDP_SDI.2.1	FDP_SDI.2.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Assignment operation completed. “more restrictive”
FDP_SDI.2.2	FDP_SDI.2.2/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Assignment operation completed. “more restrictive”
FPR_UNO.1.1	FPR_UNO.1.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Assignment operation completed. “more restrictive”
FPT_FLS.1.1	FPT_FLS.1.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Requirement itself is all the same.
FPT_TDC.1.1	FPT_TDC.1.1/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Requirement itself is all the same.
FPT_TDC.1.2	FPT_TDC.1.2/CORE	Editorial refinement operation performed for the component/element name for consistent naming convention. Assignment operation completed. “more restrictive”
FIA_ATD.1.1/AID	FIA_ATD.1.1/AID	Same
FIA_UID.2.1/AID	FIA_UID.2.1/AID	Same
FIA_USB.1.1/AID	FIA_USB.1.1/AID	Same
FIA_USB.1.2/AID	FIA_USB.1.2/AID	Assignment operation completed. “more restrictive”
FIA_USB.1.3/AID	FIA_USB.1.3/AID	Assignment operation completed. “equivalent”

[JCSPP]	ST	Rationale
FMT_MTD.1.1/JCRE	FMT_MTD.1.1/JCRE	Same
FMT_MTD.3.1/JCRE	FMT_MTD.3.1/JCRE	Same

- InstG

[JCSPP]	ST	Rationale
FDP_ITC.2.1/Installer	FDP_ITC.2.1/Installer	Same
FDP_ITC.2.2/Installer	FDP_ITC.2.2/Installer	Same
FDP_ITC.2.3/Installer	FDP_ITC.2.3/Installer	Same
FDP_ITC.2.4/Installer	FDP_ITC.2.4/Installer	Same
FDP_ITC.2.5/Installer	FDP_ITC.2.5/Installer	Same
FMT_SMR.1.1/Installer	FMT_SMR.1.1/Installer	Same
FMT_SMR.1.2/Installer	FMT_SMR.1.2/Installer	Same
FPT_FLS.1.1/Installer	FPT_FLS.1.1/Installer	Same
FPT_RCV.3.1/Installer	FPT_RCV.3.1/Installer	Assignment operation completed. “equivalent”
FPT_RCV.3.2/Installer	FPT_RCV.3.2/Installer	Assignment operation completed. “more restrictive”
FPT_RCV.3.3/Installer	FPT_RCV.3.3/Installer	Assignment operation completed. “more restrictive”
FPT_RCV.3.4/Installer	FPT_RCV.3.4/Installer	Same

- ADELG

[JCSPP]	ST	Rationale
FDP_ACC.2.1/ADEL	FDP_ACC.2.1/ADEL	Same
FDP_ACC.2.2/ADEL	FDP_ACC.2.2/ADEL	Same
FDP_ACF.1.1/ADEL	FDP_ACF.1.1/ADEL	Same
FDP_ACF.1.2/ADEL	FDP_ACF.1.2/ADEL	Same
FDP_ACF.1.3/ADEL	FDP_ACF.1.3/ADEL	Same
FDP_ACF.1.4/ADE	FDP_ACF.1.4/ADE	Same

[JCSPP]	ST	Rationale
L	L	
FDP_RIP.1.1/ADEL	FDP_RIP.1.1/ADEL	Same
FMT_MSA.1.1/ADEL	FMT_MSA.1.1/ADEL	Same
FMT_MSA.3.1/ADEL	FMT_MSA.3.1/ADEL	Same
FMT_MSA.3.2/ADEL	FMT_MSA.3.2/ADEL	Same
FMT_SMF.1.1/ADEL	FMT_SMF.1.1/ADEL	Same
FMT_SMR.1.1/ADEL	FMT_SMR.1.1/ADEL	Same
FMT_SMR.1.2/ADEL	FMT_SMR.1.2/ADEL	Same
FPT_FLS.1.1/ADEL	FPT_FLS.1.1/ADEL	Same

- RMIG

[JCSPP]	ST	Rationale
FDP_ACC.2.1/JCRMI	FDP_ACC.2.1/JCRMI	Same
FDP_ACC.2.2/JCRMI	FDP_ACC.2.2/JCRMI	Same
FDP_ACF.1.1/JCRMI	FDP_ACF.1.1/JCRMI	Same
FDP_ACF.1.2/JCRMI	FDP_ACF.1.2/JCRMI	Same
FDP_ACF.1.3/JCRMI	FDP_ACF.1.3/JCRMI	Same
FDP_ACF.1.4/JCRMI	FDP_ACF.1.4/JCRMI	Same
FDP_IFC.1.1/JCRMI	FDP_IFC.1.1/JCRMI	Same
FDP_IFF.1.1/JCRMI	FDP_IFF.1.1/JCRMI	Same
FDP_IFF.1.2/JCRMI	FDP_IFF.1.2/JCRMI	Same
FDP_IFF.1.3/JCRMI	FDP_IFF.1.3/JCRMI	Assignment operation completed. “equivalent”
FDP_IFF.1.4/JCRMI	FDP_IFF.1.4/JCRMI	Assignment operation completed. “equivalent”
FDP_IFF.1.5/JCRMI	FDP_IFF.1.5/JCRMI	Assignment operation completed. “equivalent”
FMT_MSA.1.1/EXPORT	FMT_MSA.1.1/EXPORT	Same

[JCSPP]	ST	Rationale
FMT_MSA.1.1/REM_REFS	FMT_MSA.1.1/REM_REFS	Same
FMT_MSA.3.1/JCRMI	FMT_MSA.3.1/JCRMI	Same
FMT_MSA.3.2/JCRMI	FMT_MSA.3.2/JCRMI	Same
FMT_REV.1.1/JCRMI	FMT_REV.1.1/JCRMI	Same
FMT_REV.1.2/JCRMI	FMT_REV.1.2/JCRMI	Same
FMT_SMF.1.1/JCRMI	FMT_SMF.1.1/JCRMI	Same
FMT_SMR.1.1/JCRMI	FMT_SMR.1.1/JCRMI	Same
FMT_SMR.1.2/JCRMI	FMT_SMR.1.2/JCRMI	Same

- ODELG

[JCSPP]	ST	Rationale
FDP_RIP.1.1/ODEL	FDP_RIP.1.1/ODEL	Same
FPT_FLS.1.1/ODEL	FPT_FLS.1.1/ODEL	Same

- CarG

[JCSPP]	ST	Rationale
FCO_NRO.2.1/CM	FCO_NRO.2.1/CM	Same
FCO_NRO.2.2/CM	FCO_NRO.2.2/CM	Same
FCO_NRO.2.3/CM	FCO_NRO.2.3/CM	Assignment operation completed. “more restrictive”
FDP_IFC.2.1/CM	FDP_IFC.2.1/CM	Refinement operation performed. “more restrictive”
FDP_IFC.2.2/CM	FDP_IFC.2.2/CM	Same
FDP_IFF.1.1/CM	FDP_IFF.1.1/CM	Assignment operation completed. “more restrictive”
FDP_IFF.1.2/CM	FDP_IFF.1.2/CM	Assignment operation completed. “more restrictive”
FDP_IFF.1.3/CM	FDP_IFF.1.3/CM	Assignment operation completed. “equivalent”
FDP_IFF.1.4/CM	FDP_IFF.1.4/CM	Assignment operation completed. “equivalent”
FDP_IFF.1.5/CM	FDP_IFF.1.5/CM	Assignment operation completed. “equivalent”
FDP_UIT.1.1/CM	FDP_UIT.1.1/CM	Selection operation completed. “more restrictive”

[JCSPP]	ST	Rationale
FDP_UIT.1.2/CM	FDP_UIT.1.2/CM	Same
FIA_UID.1.1/CM	FIA_UID.1.1/CM	Assignment operation completed. “more restrictive”
FIA_UID.1.2/CM	FIA_UID.1.2/CM	Same
FMT_MSA.1.1/CM	FMT_MSA.1.1/CM	Selection and Assignment operation completed. “more restrictive”
FMT_MSA.3.1/CM	FMT_MSA.3.1/CM	Same
FMT_MSA.3.2/CM	FMT_MSA.3.2/CM	Assignment operation completed. “equivalent”
FMT_SMF.1.1/CM	FMT_SMF.1.1/CM	Assignment operation completed. “more restrictive”
FMT_SMR.1.1/CM	FMT_SMR.1.1/CM	Assignment operation completed. “more restrictive”
FMT_SMR.1.2/CM	FMT_SMR.1.2/CM	Same
FTP_ITC.1.1/CM	FTP_ITC.1.1/CM	Same
FTP_ITC.1.2/CM	FTP_ITC.1.2/CM	Same
FTP_ITC.1.3/CM	FTP_ITC.1.3/CM	Same

- CMGRG & SCPG

All Security Functional Requirement in these two groups are additionally required due to the scope of the TOE in the ST, which includes the Smart Card Platform and GP.

- Security Assurance Requirements

All Security Assurance Requirements in the ST are same as stated in [JCSPP].

3 Security Aspects

[ST] Application note: The ST accepts all security aspects of the [JCSPP] and simply restates them for easy understanding of security concerns for readers.

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

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

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

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

3.1 Confidentiality

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

	System data. Java Card System data includes the data managed by the Java Card RE, the Java Card VM and the internal data of Java Card platform API classes as well.

3.2 Integrity

#.INTEG-APPLI-CODE	Application code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to the memory zone where executable code is stored. In post-issuance application loading, this threat also concerns the modification of application code in transit to the card.
#.INTEG-APPLI-DATA	Application data must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain unauthorized write access to application data. In post-issuance application loading, this threat also concerns the modification of application data contained in a package in transit to the card. For instance, a package contains the values to be used for initializing the static fields of the package.
#.INTEG-JCS-CODE	Java Card System code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to executable code.
#.INTEG-JCS-DATA	Java Card System data must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to Java Card System data. Java Card System data includes the data managed by the Java Card RE, the Java Card VM and the internal data of Java Card API classes as well.

3.3 Unauthorized execution

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

	(2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code. Note that execute access to native code of the Java Card System and applications is the concern of #.NATIVE.
#.FIREWALL	The Firewall shall ensure controlled sharing of class instances[1], and isolation of their data and code between packages (that is, controlled execution contexts) as well as between packages and the JCRE context. An applet shall not read, write, compare a piece of data belonging to an applet that is not in the same context, or execute one of the methods of an applet in another context without its authorization.
#.NATIVE	Because the execution of native code is outside of the JCS TSF scope, it must be secured so as to not provide ways to bypass the TSFs of the JCS. Loading of native code, which is as well outside those TSFs, is submitted to the same requirements. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

3.4 Bytecode verification

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

3.4.1 CAP file verification

Bytecode verification includes checking at least the following properties: (3) bytecode instructions represent a legal set of instructions used on the Java Card platform; (4) adequacy of bytecode operands to bytecode semantics; (5) absence of operand stack overflow/underflow; (6) control flow confinement to the current method (that is, no control jumps to outside the method); (7) absence of illegal data conversion and reference forging; (8) enforcement of the private/public access modifiers for class and class members; (9) validity of any kind of reference used in the bytecode (that is, any pointer to a bytecode, class, method, object, local variable, etc actually points to the beginning of piece of data of the expected kind); (10) enforcement of rules for binary compatibility (full details are given in [JCVM22], [JVM], [JCBV]). The actual set of checks performed by the verifier is implementation-dependent, but shall at

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

least enforce all the “must clauses” imposed in [JCVM22] on the bytecode and the correctness of the CAP files’ format.

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

Another important aspect to be considered about bytecode verification and application downloading is, first, the assurance that every package required by the loaded applet is indeed on the card, in a binary-compatible version (binary compatibility is explained in [JCVM22] §4.4), second, that the export files used to check and link the loaded applet have the corresponding correct counterpart on the card.

3.4.2 Integrity and authentication

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

Alternatively, the card itself may include a verifier and perform the checks prior to the effective installation of the applet or provide means for the bytecode to be verified dynamically. On-card bytecode verifier is out of the scope of the [JCSPP].

3.4.3 Linking and verification

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

3.5 Card management

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

#.INSTALL	(1) The TOE must be able to return to a safe and consistent state when the installation of a package or an applet fails or be cancelled (whatever the reasons). (2) Installing an applet must have no effect on the code and data of already installed applets. The installation procedure should not be used to bypass the TSFs. In short, it is an atomic operation, free of harmful effects on the state of the other applets. (3) The procedure of loading and installing a package shall ensure its integrity and authenticity.
#.SID	(1) Users and subjects of the TOE must be identified. (2) The identity of sensitive users and subjects associated with administrative and privileged roles must be particularly protected; this concerns the Java Card RE, the applets registered on the card, and especially the default applet and the currently selected applet (and all other active applets in Java Card System 2.2.x). A change of identity, especially standing for an administrative role (like an applet impersonating the Java Card RE), is a severe violation of the Security Functional Requirements (SFR). Selection controls the access to any data exchange between the TOE and the CAD and therefore, must be protected as well. The loading of a package or any exchange of data through the APDU buffer (which can be accessed by any applet) can lead to disclosure of keys, application code or data, and so on.
#OBJ-DELETION	(1) Deallocation of objects should not introduce security holes in the form of references pointing to memory zones that are not longer in use, or have been reused for other purposes. Deletion of collection of objects should not be maliciously used to circumvent the TSFs. (2) Erasure, if deemed successful, shall ensure that the deleted class instance is no longer accessible.
#DELETION	(1) Deletion of installed applets (or packages) should not introduce security holes in the form of broken references to garbage collected code or data, nor should they alter integrity or confidentiality of remaining applets. The deletion procedure should not be maliciously used to bypass the TSFs. (2) Erasure, if deemed successful, shall ensure that any data owned by the deleted applet is no longer accessible (shared objects shall either prevent deletion or be made inaccessible). A deleted applet cannot be selected or receive APDU commands. Package deletion shall make the code of the package no longer available for execution. (3) Power failure or other failures during the process shall be taken into account in the implementation so as to preserve the SFRs. This does not mandate, however, the process to be atomic. For instance, an interrupted deletion may result in the

	loss of user data, as long as it does not violate the SFRs. The deletion procedure and its characteristics (whether deletion is either physical or logical, what happens if the deleted application was the default applet, the order to be observed on the deletion steps) are implementation-dependent. The only commitment is that deletion shall not jeopardize the TOE (or its assets) in case of failure (such as power shortage). Deletion of a single applet instance and deletion of a whole package are functionally different operations and may obey different security rules. For instance, specific packages can be declared to be undeletable (for instance, the Java Card API packages), or the dependency between installed packages may forbid the deletion (like a package using super classes or super interfaces declared in another package).

3.6 Services

#.ALARM	The TOE shall provide appropriate feedback upon detection of a potential security violation. This particularly concerns the type errors detected by the bytecode verifier, the security exceptions thrown by the Java Card VM, or any other security-related event occurring during the execution of a TSF.
#.OPERATE	(1) The TOE must ensure continued correct operation of its security functions. (2) In case of failure during its operation, the TOE must also return to a well-defined valid state before the next service request.
#.RESOURCES	The TOE controls the availability of resources for the applications and enforces quotas and limitations in order to prevent unauthorized denial of service or malfunction of the TSFs. This concerns both execution (dynamic memory allocation) and installation (static memory allocation) of applications and packages.
#.CIPHER	The TOE shall provide a means to the applications for ciphering sensitive data, for instance, through a programming interface to low-level, highly secure cryptographic services. In particular, those services must support cryptographic algorithms consistent with cryptographic usage policies and standards.
#.KEY-MNGT	The TOE shall provide a means to securely manage cryptographic keys. This includes: (1) Keys shall be generated in accordance with specified cryptographic key generation algorithms and specified cryptographic key sizes, (2) Keys must be distributed in accordance with specified cryptographic key distribution methods, (3) Keys must be initialized before being used, (4) Keys shall be destroyed in accordance with specified

	cryptographic key destruction methods.
#.PIN-MNGT	The TOE shall provide a means to securely manage PIN objects. This includes: (1) Atomic update of PIN value and try counter, (2) No rollback on the PIN-checking function, (3) Keeping the PIN value (once initialized) secret (for instance, no clear-PIN-reading function), (4) Enhanced protection of PIN's security attributes (state, try counter…) in confidentiality and integrity.
#.SCP	The smart card platform must be secure with respect to the SFRs. Then: (1) After a power loss, RF signal loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state. (2) It does not allow the SFRs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the Java Card API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System. (3) It provides secure low-level cryptographic processing to the Java Card System. (4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism. (5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low–level control accesses (segmentation fault detection). (6) It safely transmits low-level exceptions to the TOE (arithmetic exceptions, checksum errors), when applicable. Finally, it is required that (7) the IC is designed in accordance with a well-defined set of policies and standards (for instance, those specified in [PP0035]), 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.
#.TRANSACTION	The TOE must provide a means to execute a set of operations atomically. This mechanism must not jeopardise the execution of the user applications. The transaction status at the beginning of an applet session must be closed (no pending updates).

4 Security problem definition

This chapter describes the security aspects of the TOE environment as assets to be protected, threats, organisational security policies, and assumptions.

4.1 Assets

Assets are security-relevant elements to be directly protected by the TOE. Confidentiality of assets is always intended with respect to un-trusted people or software, as various parties are involved during the first stages of the smart card product life-cycle; details are given in threats hereafter.
Assets may overlap, in the sense that distinct assets may refer (partially or wholly) to the same piece of information or data. For example, a piece of software may be either a piece of source code (one asset) or a piece of compiled code (another asset), and may exist in various formats at different stages of its development (digital supports, printed paper). This separation is motivated by the fact that a threat may concern one form at one stage, but be meaningless for another form at another stage.
The assets to be protected by the TOE are listed below. They are grouped according to whether it is data created by and for the user (User data) or data created by and for the TOE (TSF data). For each asset it is specified the kind of dangers that weigh on it.

4.1.1 User Data

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

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

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

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

D.PIN
Any end-user's PIN.

To be protected from unauthorized disclosure and modification.

4.1.2 TSF Data

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

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

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

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

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

D.OS_DATA
The platform data including the GP registry and other OS data.
To be protected from unauthorized disclosure and modification.
[ST] Application note:
The ST author considered this additional TSF data as security-relevant due to the scope of the TOE including GP as Card Manager.

D.OS_CODE
The platform code.
To be protected from unauthorized disclosure and modification.
[ST] Application note:
The ST author considered this additional TSF data as security-relevant due to the scope of the TOE including GP as Card Manager.

D.ISD_KEYS
The GP Issuer Security Domain keys.
To be protected from unauthorized disclosure and modification.
[ST] Application note:
The ST author considered this additional TSF data as security-relevant due to the scope of the TOE including GP as Card Manager.

D.SD_KEYS
The GP additional Security Domain keys.
To be protected from unauthorized disclosure and modification.
[ST] Application note:
The ST author considered this additional TSF data as security-relevant due to the scope of the TOE including GP as Card Manager.

4.2 Threats

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

4.2.1 Confidentiality

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

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

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

4.2.2 Integrity

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

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

T.INTEG-APPLI-DATA

The attacker executes an application to alter (part of) another application's data. See #.INTEG-APPLI-DATA for details.
Directly threatened asset(s): D.APP_I_DATA, D.PIN and D.APP_KEYs.

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

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

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

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

4.2.3 Identity usurpation

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

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

4.2.4 Unauthorized execution

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

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

T.EXE-CODE-REMOTE
The attacker performs an unauthorized remote execution of a method from the CAD. See #.EXE-APPLI-CODE for details.
Directly threatened asset(s): D.APP_CODE.
[JCSPP] Application note:
This threat concerns version 2.2.x of the Java Card RMI, which allow external users (that is, other than on-card applets) to trigger the execution of code belonging to an on-card applet. On the contrary, T.EXE-CODE.1 is restricted to the applets under the TSF.

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

4.2.5 Denial of service

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

4.2.6 Card management

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

T.INSTALL
The attacker fraudulently installs post-issuance of an applet on the card. This concerns either the installation of an unverified applet or an attempt to induce a malfunction in the TOE through the installation process. See #.INSTALL for details.
Directly threatened asset(s): D.SEC_DATA (any other asset may be jeopardized should this attack succeed, depending on the virulence of the installed application).

4.2.7 Services

T.OBJ-DELETION
The attacker keeps a reference to a garbage collected object in order to force the TOE to execute an unavailable method, to make it to crash, or to gain access to a

memory containing data that is now being used by another application. See #.OBJ-DELETION for further details.
Directly threatened asset(s): D.APP_C_DATA, D.APP_I_DATA and D.APP_KEYs.

4.2.8 Miscellaneous

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

4.2.9 Additional threats

Following threats apply to more generic attacks on the Smart Card Platform and GP software. Threat agent does not use application but observes OS behavior or uses GP specific commands.

T.ACCESS
Unauthorized access to sensitive information stored in memories in order to disclose or to corrupt the TOE data. This includes any consequences of bad or incorrect user authentication by the TOE.
[ST] Application note:
This threat addresses several methods of attacks:
- An attacker may determine Application data and GP data through observation of the results of repetitive insertion of selected data
- An attacker may penetrate on-card security through reuse of a completed (or partially completed) operation by an authorized user
- An attacker may search the entire user-accessible data space to identify GP and Application data such as PINs by example if CVM option is selected
- An attacker may defeat the card's Security Functions through a cryptographic attack against the algorithm or through a brute-force attack on the function inputs
- An attacker may exploit commands, particularly test and debug commands, which were necessary for another part of the card life cycle but are not presently allowed, to expose GP data or sensitive Application data.

T.OS_OPERATE
Modification of the correct Software behavior by unauthorized use of TOE or use of incorrect or unauthorized instructions or commands or sequence of commands, in order to obtain an unauthorized execution of the TOE code.
[ST] Application note:
- An attacker may determine security relevant information cause the card to malfunction or otherwise compromise security through introduction of invalid inputs
- An attacker may force the card into an insecure Life Cycle state through

inappropriate termination of selected operations.

T.LEAKAGE
An attacker may exploit information which is leaked from the TOE during usage of the Smart Card in order to disclose the Software behavior and Application Data handling (TSF data or User data). No direct contact with the Smart Card Internals is required here. Leakage may occur through emanations, variations in power consumption, I/O characteristics, clock frequency, or by changes in processing time requirements. One example is the Differential Power Analysis (DPA).
[ST] Application note:
- An attacker may exploit GP data that is leaked from the card during normal usage
- An attacker may observe multiple uses of resources or services and, by linking these observations, deduce information that that may reveal GP or Application data.

T.FAULT
An attacker may cause a malfunction of TSF by applying environmental stress in order to (1) deactivate or modify security features or functions of the TOE or (2) deactivate or modify security functions of the Smart Card. This may be achieved by operating the Smart Card outside the normal operating conditions.
[ST] Application note:
- An attacker may induce errors in GP data through exposure of the card to environmental stress
- An attacker may perform simultaneous attacks with the result that the card's Security Functions become unstable or some part of the GP data is degraded resulting in exposure of GP data or sensitive Application data.

T.RND
Deficiency of Random Numbers.
An attacker may predict or obtain information about random numbers generated by the TOE security service for instance because of a lack of entropy of the random numbers provided.
An attacker may gather information about the random numbers produced by the TOE security service. Because unpredictability is the main property of random numbers this may be a problem in case they are used to generate cryptographic keys. Here the attacker is expected to take advantage of statistical properties of the random numbers generated by the TOE. Malfunctions or premature ageing are also considered which may assist in getting information about random numbers.

4.2.10 Compatibility statement of threats

Threats in [ICST] are all based on the PP [PP0035] except for one additional threat. The relevant threats of the [ICST] are not contradictory to those of the composite ST.

[ICST]	composite ST
BSI.T.Leak-Inherent	T.PHYSICAL, T.LEAKAGE These threats address inherent information leakage such as DPA.
BSI.T.Phys-Probing	T.PHYSICAL This threat addresses physical probing attack.

[ICST]	composite ST
BSI.T.Malfunction	T.FAULT This threat addresses malfunction due to environmental stress.
BSI.T.Phys-Manipulation	T.PHYSICAL This threat addresses physical manipulation.
BSI.T.Leak-Forced	T.PHYSICAL, T.OS_OPERATE, T.FAULT These threats address forced information leakage which is not inherent but caused by the attacker.
BSI.T.Abuse-Func	T.ACCESS This threat addresses abuse of functionality of the TOE.
BSI.T.RND	T.RND This threat addresses deficiency of random numbers.
AUG4.T.Mem-Access	T.ACCESS This threat addresses memory access violation.

4.3 Organisational security policies

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

4.3.1 OSPs from [JCSPP]

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

OSP.DELETION
Deletion of applets through the card manager shall be secure. See #.DELETION for details.
[ST] Application note:
Originally this organizational security policy was addressed under an assumption A.DELETION in the [JCSPP]. The TOE includes the Card Manager to delete applets securely, thus the ST author moved the related assumption to the section of the security organizational policy to address this security aspects properly.

4.3.2 Additional OSPs

OSP.ROLES
The TOE shall recognize the following roles associated with:
- Card Administrator,
- Application Provider
- End-user (Card Holder).

OSP.INITIAL_LIFECYCLE_STATES

Card shall be moved in OP_READY state before any GP function or service is used.
Card shall be issued to Cardholders with the card set to SECURED life cycle state.
A security domain shall be moved into the PERSONALIZED life cycle state before any security domain User or Application uses the services of that Security Domain.

OSP.CARD_ADMINISTRATOR_PRE-APPROVAL
Only the Card Administrator shall be allowed to perform Card Content Management Functions (CCMFs).

OSP.APPLICATION_PROVIDER_PRE-APPROVAL
The Application Provider allows the Card Administrator to perform CCMFs for its own Applications as well as personalizing and managing some Application(s) specific data (or keys).

OSP.LOAD_FILE_VERIFICATION
Integrity and authenticity of the Load File shall be verified and shall always be carried out successfully prior to Application Load File installation. This shall take place on-card.

OSP.APPLICATION_CODE_VERIFICATION
Byte code verification and other forms of Application Code Verification is a requirement and shall always be carried out successfully prior to Application Load File on-card installation. This shall take place off-card. Application Code Verification shall at least include the algorithms necessary to establish that the Application would pass all omitted runtime checks.

OSP.SECURE_COMMUNICATION
Only the minimum security requirements for GP commands as defined by [VGP] are required.

OSP.CARDHOLDER_VERIFICATION
A Cardholder Verification method common to several Applications is required. Applications may also use Application-specific Cardholder Verification methods.

OSP.PROD_PROCESS
Procedures ensure protection of the TOE material/information that is stored outside the TOE and used for TOE initialization and personalization, to maintain integrity and confidentiality of the TOE and of its manufacturing and test operations to prevent any possible copy, modification, retention, theft or unauthorized use).
Appropriate functionality testing of the TOE is used TOE initialization and personalization phases.

OSP.CRYPTO
The TOE shall provide the following specific cryptographic functionality to the application provider:

- Data Encryption Standard (DES),
- Triple Data Encryption Standard (3DES),

- Advanced Encryption Standard (AES),
- Elliptic Curves Cryptography on GF(p),
- Secure Hashing (SHA-1, SHA-256),
- Rivest-Shamir-Adleman (RSA, RSA-CRT),
- Prime Number Generation.
- Elliptic Curve Digital Signature Algorithm (ECDSA)
- SEED

4.3.3 Compatibility statement of OSPs

OSPs in [ICST] are all based on the PP [PP0035] except for one additional OSP. The relevant OSPs of the [ICST] are not contradictory to those of the composite ST.

[ICST]	composite ST
BSI.P.Process-TOE	OSP.PROD_PROCESS This OSP is related to manufacturing and development of the TOE, this addresses protection during TOE development and production.
AUG1.P.Add-Functions	OSP.CRYPTO This OSP addresses specific cryptographic functionality provided by TOE.

4.4 Assumptions

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

4.4.1 Assumptions from [JCSPP]

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

A.VERIFICATION
All the bytecodes are verified at least once, before the loading, before the installation or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time.

4.4.2 Additional assumptions

A.KEY_MANAGEMENT
It is assumed that cryptographic keys, which are stored outside the TOE and which are used for secure communication and authentication between Smart Card and terminals are protected in their own (off-card) storage environment.
[ST] Application note:

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

A.CVM
It is assumed that the CVM values are generated maintained and used off card in a secure manner during personalization phases.
It is assumed that the Card Holder keeps his personal code secret.

A.ACTORS
It is assumed that the Card Administrator is the sole Application Provider and also plays the roles of Application Loader and Verification Authority.

4.4.3 Compatibility statement of assumptions

Assumptions in [ICST] are all based on the PP [PP0035]. The relevant assumptions of the [ICST] are not contradictory to those of the composite ST.

[ICST]	composite ST
BSI.A.Process-Sec-IC	Assumptions in [ICST] are related to the development and manufacturing phases and are covered by the OSP OSP.PROD_PROCESS therefore considered as being fulfilled automatically.
BSI.A.Plat-Appl	
BSI.A.Resp-Appl	

5 Security objectives

5.1 Security objectives for the TOE

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

5.1.1 Identification

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

5.1.2 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.

5.1.3 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.
[JCSPP] 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.REMOTE
The TOE shall provide restricted remote access from the CAD to the services implemented by the applets on the card. This particularly concerns the Java Card RMI services introduced in version 2.2.x of the Java Card platform.

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.

5.1.4 Object deletion

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

5.1.5 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.
[JCSPP] 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).

5.1.6 Reassignment

O.CARD-MANAGEMENT
The TOE 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 of the TOE is an application with specific rights, which is responsible for the administration of the smart card. This component will in practice be tightly connected with the rest part of the TOE, which in turn shall very likely rely on the card manager for the effective enforcing of some of its security functions. Typically the card manager of the TOE shall be in charge of the life cycle of the whole card, as well as that of the installed applications (applets). The card manager of the TOE should prevent that card content management (loading, installation, deletion) is carried out, for instance, at invalid states of the card or by non-authorized actors. It shall also enforce security policies established by the card issuer.
[ST] Application note:
Originally this security objective was addressed under OE.CARD-MANAGEMENT in the [JCSPP]. The TOE includes the Card Manager to manage applets securely, thus the ST author reassigned the related security objective for the operational environment to the security objective for the TOE to address this security aspects properly.

O.SCP.IC
The TOE shall provide all IC security features against physical attacks.
This security objective for the TOE 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.
[ST] Application note:
Originally this security objective was addressed under OE.SCP.IC in the [JCSPP]. The TOE includes the SCP, especially the certified IC chip, and TOE is a subject to the composite evaluation against the [COMP-EVAL], thus the ST author reassigned the related security objective for the operational environment to the security objective for the TOE to address this security aspects properly.

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 TOE must allow the TOE itself to eventually complete the interrupted operation successfully, or recover to a consistent and secure state.
This security objective for the TOE refers to the security aspect #.SCP(1): The TOE 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 TOE will allow the TOE itself on the next power up to either complete the interrupted operation or revert to a secure state.
[ST] Application note:
Originally this security objective was addressed under OE.SCP.RECOVERY in the [JCSPP]. The TOE includes the SCP, especially the certified IC chip, and TOE is a subject to the composite evaluation against the [COMP-EVAL], thus the ST author reassigned the related security objective for the operational environment to the security objective for the TOE to address this security aspects properly.

O.SCP.SUPPORT
The TOE shall support the TSFs of the TOE itself.
This security objective for the TOE 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).
[ST] Application note:
Originally this security objective was addressed under OE.SCP.SUPPORT in the [JCSPP]. The TOE includes the SCP, especially the certified IC chip, and TOE is a subject to the composite evaluation against the [COMP-EVAL], thus the ST author reassigned the related security objective for the operational environment to the security objective for the TOE to address this security aspects properly.

5.1.7 Additional security objectives for the TOE

O.PROTECT_DATA

The TOE shall ensure that sensitive information stored in memories is protected against unauthorized disclosure and any corruption or unauthorized modification.
Moreover, the TOE shall ensure that sensitive information stored in memories is protected against unauthorized access.
The TOE has to provide appropriate security mechanisms to avoid fraudulent access to any sensitive data, such as passwords, cryptographic keys or authentication data.
This is obvious for secret information, but also applies to access controlled information

O.OS_OPERATE
The TOE must ensure continued correct operation of its security functions.
Especially, the TOE must prevent the unauthorized use of TOE or use of incorrect or unauthorized instructions or commands or sequence of commands.
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.

O.SIDE_CHANNEL
The TOE must provide protection against disclosure of confidential data (User Data or TSF data) stored and/or processed in the Smart Card IC:
- By measurement and analysis of the shape and amplitude of signals (for example on the power, clock, or I/O lines),
- By measurement and analysis of the time between events found by measuring signals (for example on the power, clock, or I/O lines).
Especially, the software must be designed to avoid interpretations of signals extracted, intentionally or not, from the hardware part of the TOE (for instance, Power Supply, Electro Magnetic emissions).

O.FAULT_PROTECT
The TOE must ensure its correct operation even outside the normal operating conditions where reliability and secure operation has not been proven or tested. This is to prevent errors. The environmental conditions may include voltage, clock frequency, temperature, or external energy fields that can be applied on all interfaces of the TOE (physical or electrical).

O.RND
The TOE will ensure the cryptographic quality of random number generation. For instance random numbers shall not be predictable and shall have sufficient entropy.
The TOE will ensure that no information about the produced random numbers is available to an attacker since they might be used for instance to generate cryptographic keys.

O.ROLES
The TOE shall recognize the following roles associated with:
- Card Administrator,
- Application Provider
- End-user (Card Holder).

O.CARD_ADMIN

The TOE shall provide the Card Administrator with mean to perform secure CCMFs.

O.APPLICATION_PROVIDER_PRE-APPROVAL
The TOE shall allow the Application Provider to allow the Card Administrator to perform CCMFs for its own Applications as well as personalizing and managing some Application(s) specific data (or keys).

O.LOAD_FILE_VERIFICATION
The TOE shall provide means to verify the integrity and authenticity of the Load File. This verification shall always be carried out successfully prior to Application Load File installation. This shall take place on-card.

O.APPLICATION_CODE_VERIFICATION
The TOE shall provide means to confirm that byte code and other forms of Application Code Verification has been performed prior to Application Load File on-card installation. This shall take place off-card.
[ST] Application note:
This objective is related to P.APPLICATION_CODE_VERIFICATION. In this case the verification is performed outside the TOE but the TOE is required to provide means to verify that the operation has been performed

O.SECURE_COMM
The TOE shall provide secure communication protocol as defined by [VGP].

O.CARDHOLDER_VERIFICATION
The TOE shall provide Cardholder Verification method in accordance with the P.CARDHOLDER_VERIFICATION policy.

5.1.8 Compatibility statement of security objectives for the TOE

Security objectives for the TOE in [ICST] are all based on the PP [PP0035] except for two additional security objectives. The relevant security objectives for the TOE of the [ICST] are not contradictory to those of the composite ST.

[ICST]	composite ST
BSI.O.Leak-Inherent	O.SCP.IC, O.SIDE_CHANNEL These security objectives address inherent information leakage such as DPA.
BSI.O.Phys-Probing	O.SCP.IC This security objective addresses physical probing attack.
BSI.O.Malfunction	O.FAULT_PROTECT, O.OS_OPERATE This security objective addresses malfunction due to environmental stress.
BSI.O.Phys-Manipulation	O.SCP.IC This security objective addresses physical manipulation.

[ICST]	composite ST
BSI.O.Leak-Forced	O.SCP.IC, O.OS_OPERATE, O.FAULT_PROTECT These security objective address forced information leakage which is not inherent but caused by the attacker.
BSI.O.Abuse-Func	There is no direct relevant security objective. O.SCP.IC, O.OS_PROTECT These security objectives address abuse of functionality of the TOE indirectly.
BSI.O.Identification	There is no relevant security objective for the TOE. Instead, this is addressed by OE.PROD_PROCESS. This OSP is related to manufacturing and development of the TOE, this addresses protection during TOE development and production.
BSI.O.RND	O.RND This security objective addresses deficiency of random numbers.
AUG1.O.Add-Functions	O.CIPHER This security objective addresses specific cryptographic functionality provided by TOE.
AUG4.O.Mem-Access	O.PROTECT_DATA This security objective addresses memory access violation.

5.2 Security objectives for the operational environment

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

5.2.1 Security objectives for the operational environment from [JCSPP]

OE.APPLET
No applet loaded post-issuance shall contain native methods.

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

5.2.2 Additional Security objectives for the operational environment

OE.ACTORS
The Card Administrator is the sole Application Provider and also plays the roles of Application Loader and Verification Authority.

OE.INITIAL_LIFECYCLE_STATES

After Card manufacturing and initialization, the card administrator shall move the Card in the OP_READY state before any GP function or service is used.
The card Issuer shall issue the card to the Cardholders with the card set to SECURED life cycle state.
A security domain shall be moved into the PERSONALIZED life cycle state before any security domain User or Application uses the services of that Security Domain.

OE.PROD_PROCESS
Appropriate functionality testing of the TOE shall be used in during initialization, personalization and other operations before Issuance.
During these operations, security procedures shall be used to maintain confidentiality and integrity of the TOE manufacturing and test data.

OE.KEY_MANAGEMENT
During the TOE usage, the terminal or system in interaction with the TOE shall ensure the protection of their own keys by operational means and/or procedures.

OE.CVM
The CVM values shall be generated, maintained and used off card in a secure manner during personalization phases.
The Card Holder keeps his personal code secret.

5.2.3 Compatibility statement of security objectives for the operational environment

Security objectives for the operational environment in [ICST] are all based on the PP [PP0035]. The relevant security objectives for the operational environment of the [ICST] are not contradictory to those of the composite ST.

[ICST]	composite ST
BSI.OE.Process-Sec-IC	Security objectives for the operational environment in [ICST] are related to the development and manufacturing phases and are covered by OE.PROD_PROCESS therefore considered as being fulfilled automatically.
BSI.OE.Plat-Appl	
BSI.OE.Resp-Appl	

5.3 Security objectives rationale

5.3.1 Threats

5.3.1.1 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 ST 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.

5.3.1.2 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.
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 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.
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.

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 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.

5.3.1.3 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.

5.3.1.4 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.EXE-CODE-REMOTE
The O.REMOTE security objective contributes to prevent the invocation of a method

that is not supposed to be accessible from outside the card.

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).

5.3.1.5 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. 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.

5.3.1.6 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.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.

5.3.1.7 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.

5.3.1.8 Miscellaneous

T.PHYSICAL
This threat is covered by O.SCP.IC. Physical protections rely on the underlying platform and are therefore an environmental issue.

5.3.1.9 Additional threats

T.ACCESS
This threat is covered by O.PROTECT_DATA which addresses the protection of data stored in the TOE from unauthorized disclosure and any corruption or unauthorized modification.

T.OS_OPERATE
This threat is covered by O.OS_OPERATE which requires to ensure continue correct operation of the TOE regarding incorrect usage or power loss with recovery of a secure state.

T.LEAKAGE
This threat is covered by O.SIDE_CHANNEL that requires protection against disclosure of confidential data by measurements and analysis of signals emitted by the TOE.

T.FAULT
This threat is covered by O.FAULT_PROTECT that requires correct behavior of the TOE when operating outside the normal range.

T.RND
This threat is covered by O.RND that requires the cryptographic quality of random number generation.

5.3.2 Organisational security policies

5.3.2.1 OSPs from [JCSPP]

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

OSP.DELETION
This policy is upheld by the environmental objective O.CARD-MANAGEMENT which controls the access to card management functions such as deletion of applets.

5.3.2.2 Additional OSPs

OSP.ROLES

This policy is upheld by O.ROLES which requires roles to be recognized in the TOE.

OSP.INITIAL_LIFECYCLE_STATES
This policy is upheld by OE.INITIAL_LIFECYCLE_STATES which specifies appropriate roles to move card state.

OSP.CARD_ADMINISTRATOR_PRE-APPROVAL
This policy is upheld by O.CARD_ADMIN which requires the TOE to provide means to perform secure CCMFs to Card Administrator.

OSP.APPLICATION_PROVIDER_PRE-APPROVAL
This policy is upheld by O.APPLICATION_PROVIDER_PRE-APPROVAL which requires the TOE to provide means for the Application Provider to allow the Card Administrator to perform CCMFs for its own Applications.

OSP.LOAD_FILE_VERIFICATION
This policy is upheld by O.LOAD_FILE_VERIFICATION which requires the TOE to provide means to verify the integrity and authenticity of the Load File.

OSP.APPLICATION_CODE_VERIFICATION
This policy is upheld by O.APPLICATION_CODE_VERIFICATION which requires the TOE to provide means to verify that byte code and other forms of application code verification has been performed.

OSP.SECURE_COMMUNICATION
This policy is upheld by O.SECURE_COMM which requires the TOE to provide secure communication protocol as defined by [VGP].

OSP.CARDHOLDER_VERIFICATION
This policy is upheld by O.CARDHOLDER_VERIFICATION which requires the TOE to provide Cardholder Verification method in accordance with the policy.

OSP.PROD_PROCESS
This policy is upheld by OE.PROD_PROCESS which specifies manufacturing and development of the TOE.

OSP.CRYPTO
This policy is upheld by O.CIPHER which requires the TOE to provide specific cryptographic functionality to the application provider.

5.3.3 Assumptions

5.3.3.1 Assumptions from [JCSPP]

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

A.VERIFICATION
This assumption is upheld by the security objective for 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.

5.3.3.2 Additional Assumptions

A.KEY_MANAGEMENT
This assumption is upheld by the security objective for the operational environment OE.KEY_MANAGEMENT which guarantees secure management of cryptographic keys which is shared with the TOE but is not under the TOE control, may compromise the security of the TOE.

A.CVM
This assumption is upheld by the security objective for the operational environment OE.CVM which guarantees that the CVM values are generated maintained and used off card in a secure manner during personalization phases and the Card Holder keeps his personal code secret.

A.ACTORS
This assumption is upheld by the security objective for the operational environment OE.ACTORS which guarantees that the Card Administrator is the sole Application Provider and also plays the roles of Application Loader and Verification Authority.

5.3.4 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 5.3.1.1
T.CONFID-JCS-CODE	OE.VERIFICATION, O.CARD-MANAGEMENT, O.NATIVE	Section 5.3.1.1
T.CONFID-JCS-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT,	Section 5.3.1.1

Threats	Security Objectives	Rationale
	OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM	
T.INTEG-APPLI-CODE	O.CARD-MANAGEMENT, OE.VERIFICATION, O.NATIVE	Section 5.3.1.2
T.INTEG-APPLI-CODE.LOAD	O.LOAD, O.CARD-MANAGEMENT	Section 5.3.1.2
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	Section 5.3.1.2
T.INTEG-APPLI-DATA.LOAD	O.LOAD, O.CARD-MANAGEMENT	Section 5.3.1.2
T.INTEG-JCS-CODE	O.CARD-MANAGEMENT, OE.VERIFICATION, O.NATIVE	Section 5.3.1.2
T.INTEG-JCS-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD-MANAGEMENT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM	Section 5.3.1.2
T.SID.1	O.CARD-MANAGEMENT, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.INSTALL, O.SID	Section 5.3.1.3
T.SID.2	O.SCP.RECOVERY, O.SCP.SUPPORT, O.SID, O.OPERATE, O.FIREWALL, O.INSTALL	Section 5.3.1.3

Threats	Security Objectives	Rationale
T.EXE-CODE.1	OE.VERIFICATION, O.FIREWALL	Section 5.3.1.4
T.EXE-CODE.2	OE.VERIFICATION	Section 5.3.1.4
T.EXE-CODE-REMOTE	O.REMOTE	Section 5.3.1.4
T.NATIVE	OE.VERIFICATION, OE.APPLET, O.NATIVE	Section 5.3.1.4
T.RESOURCES	O.INSTALL, O.OPERATE, O.RESOURCES O.SCP.RECOVERY, O.SCP.SUPPORT	Section 5.3.1.5
T.DELETION	O.DELETION, O.CARD-MANAGEMENT	Section 5.3.1.6
T.INSTALL	O.INSTALL, O.LOAD, O.CARD-MANAGEMENT	Section 5.3.1.6
T.OBJ-DELETION	O.OBJ-DELETION	Section 5.3.1.7
T.PHYSICAL	O.SCP.IC	Section 5.3.1.8
T.ACCESS	O.PROTECT_DATA	Section 5.3.1.9
T.OS_OPERATE	O.OS_OPERATE	Section 5.3.1.9
T.LEAKAGE	O.SIDE_CHANNEL	Section 5.3.1.9
T.FAULT	O.FAULT_PROTECT	Section 5.3.1.9
T.RND	O.RND	Section 5.3.1.9

Table5. Threats and Security Objectives - Coverage

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

Security Objectives	Threats
	T.NATIVE
O.OPERATE	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES
O.REALLOCATION	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.RESOURCES	T.RESOURCES
O.ALARM	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA
O.CIPHER	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.KEY-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.PIN-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.REMOTE	T.EXE-CODE-REMOTE
O.TRANSACTION	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA
O.OBJ-DELETION	T.OBJ-DELETION
O.DELETION	T.DELETION
O.LOAD	T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA.LOAD, T.INSTALL
O.INSTALL	T.SID.1, T.SID.2, T.RESOURCES, T.INSTALL
O.CARD-MANAGEMENT	T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA, T.INTEG-APPLI-DATA.LOAD, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.SID.1, T.DELETION,T.INSTALL
O.SCP.IC	T.PHYSICAL
O.SCP.RECOVERY	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA,

Security Objectives	Threats
	T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES
O.SCP.SUPPORT	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES
O.PROTECT_DATA	T.ACCESS
O.OS_OPERATE	T.OS_OPERATE
O.SIDE_CHANNEL	T.LEAKAGE
O.FAULT_PROTECT	T.FAULT
O.RND	T.RND
OE.APPLET	T.NATIVE
OE.VERIFICATION	T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-DATA, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.EXE-CODE.1, T.EXE-CODE.2, T.NATIVE

Table6. Security Objectives and Threats - Coverage

OSPs	Security Objectives	Rationale
OSP.VERIFICATION	OE.VERIFICATION	Section 5.3.2.1
OSP.DELETION	O.CARD-MANAGEMENT	Section 5.3.2.1
OSP.ROLES	O.ROLES	Section 5.3.2.2
OSP.INITIAL_LIFECYCLE_STATES	OE.INITIAL_LIFECYCLE_STATES	Section 5.3.2.2
OSP.CARD_ADMINISTRATOR_PRE-APPROVAL	O.CARD_ADMIN	Section 5.3.2.2
OSP.APPLICATION_PROVIDER_PRE-APPROVAL	O.APPLICATION_PROVIDER_PRE-APPROVAL	Section 5.3.2.2
OSP.LOAD_FILE_VERIFICATION	O.LOAD_FILE_VERIFICATION	Section 5.3.2.2
OSP.APPLICATION_CODE_VERIFICATION	O.APPLICATION_CODE_VERIFICATION	Section 5.3.2.2
OSP.SECURE_COMMUNICATION	O.SECURE_COMM	Section 5.3.2.2
OSP.CARDHOLDER_VERIFICATION	O.CARDHOLDER_VERIFICATION	Section 5.3.2.2
OSP.PROD_PROCESS	OE.PROD_PROCESS	Section 5.3.2.2
OSP.CRYPTO	O.CIPHER	Section 5.3.2.2

Table7. OSPs and Security Objectives- Coverage

Assumptions	Security Objectives for the Operational Environment	Rationale
A.APPLET	OE.APPLET	Section 5.3.3.1
A.VERIFICATION	OE.VERIFICATION	Section 5.3.3.1
A.KEY_MANAGEMENT	OE.KEY_MANAGEMENT	Section 5.3.3.2
A.CVM	OE.CVM	Section 5.3.3.2
A.ACTORS	OE.ACTORS	Section 5.3.3.2

Table8. Assumptions and Security Objectives for the Operational Environment - Coverage

6 Extended components definition

Application note:
The following extended component is from [PP0035] which is conformance claimed by the [ICST].

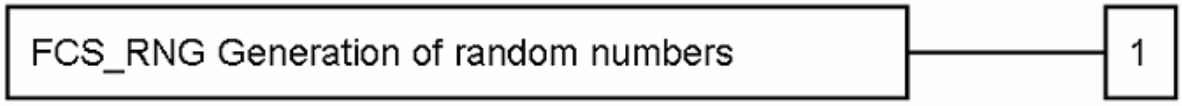
6.1 Definition of the Family FCS_RNG

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

Component leveling:



FCS_RNG.1	**Generation of random numbers requires that random numbers meet a defined quality metric.**
Management:	FCS_RNG.1 There are no management activities foreseen.
Audit:	FCS_RNG.1 There are no actions defined to be auditable.
FCS_RNG.1	Random number generation
Hierarchical to:	No other components.
Dependencies:	No dependencies.
FCS_RNG.1.1	The TSF shall provide a [selection: physical, non-physical true, deterministic, hybrid] random number generator that implements: [assignment: list of security capabilities].
FCS_RNG.1.2	The TSF shall provide random numbers that meet [assignment: a defined quality metric].
[ST] Application Note:	A physical random number generator (RNG) produces the random number by a noise source based on physical random processes. A non-physical true RNG uses a noise source based on non-physical random processes like human interaction (key strokes, mouse movement). A deterministic RNG uses a random seed to produce a pseudorandom output. A hybrid RNG combines the principles of physical and deterministic RNGs.

7 Security requirements

7.1 Security functional requirements

This section states the security functional requirements for the TOE. The TOE SFRs are arranged into groups, according to the scope defined for the TOE, CMGRG is added for the Card Manager and SCPG is added for the smart card platform.

Group	Description	Remark
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]).	[JCSPP]
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.	
Remote Method Invocation (RMIG)	The RMIG contains the security requirements for the remote method invocation feature, which provides a new protocol of communication between the terminal and the applets. This was introduced in Java Card specification version 2.2.	
Object deletion (ODELG)	The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism. This is a Java Card specification version 2.2 feature.	
Secure carrier (CarG)	The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing, in those configurations that do not support on-card static or dynamic bytecode verification, the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.	
Card manager (CMGRG)	The CMGRG group contains the security requirements for the card manager. This group contains the security requirements for a policy for controlling access to card content management operations and for expressing card issuer security concerns. Also, this group contains the security requirements to fulfill GP specific objectives.	[JCSPP] Additions

Group	Description	Remark
Smart card Platform (SCPG)	The SCPG group contains the security requirements for the smart card platform, that is, operating system and chip that the Java Card System is implemented upon.	

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, and 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	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 §7.1.3.1.
S.APPLET	Any applet instance.
S.BCV	The bytecode verifier (BCV), which acts on behalf of the verification authority who is in charge of the bytecode verification of the packages. This subject is involved in the PACKAGE LOADING security policy defined in §7.1.7.
S.CAD	The CAD represents the actor that requests, by issuing commands to the card, for RMI services. It also plays the role of the off-card entity that communicates with the S.INSTALLER.
S.INSTALLER	The installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the loading of packages and installation of applets.
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.
S.OPEN	The central on-card administrator that owns the GlobalPlatform Registry
S.ISD	On-card entity providing support for the control, security, and communication requirements of the Card Issuer
S.SD	Supplementary Security Domain to identify an Application Provider’s or Controlling Authorities’ Security Domain

Subject	Description
	Supplementary Security Domain for Application Provider or Controlling Authority is installed on the TOE, and can be activated if necessary

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

Object	Description
O.APPLET	Any installed applet, its code and data.
O.CODE_PKG	The code of a package, including all linking information. On the Java Card platform, a package is the installation unit.
O.JAVAOBJECT	Java class instance or array. It should be noticed that KEYS, PIN, arrays and applet instances are specific objects in the Java programming language.
O.REMOTE_MTHD	A method of a remote interface.
O.REMOTE_OBJ	A remote object is an instance of a class that implements one (or more) remote interfaces. A remote interface is one that extends, directly or indirectly, the interface java.rmi.Remote ([JCAPI22]).
O.RMI_SERVICE	These are instances of the class javacardx.rmi.RMIService. They are the objects that actually process the RMI services.
O.ROR	A remote object reference. It provides information concerning: (i) the identification of a remote object and (ii) the Implementation class of the object or the interfaces implemented by the class of the object. This is the object's information to which the CAD can access.
O.CARD_CONTENT	Code, Application information, and Application data contained in the card that is under the responsibility of the OPEN e.g. Executable Load Files, Application instances, etc They are subject to Card Content Management Functions (CCMFs)
O.REGISTRY	A container of information related to Card Content management.

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

Information	Description
I.APDU	Any APDU sent to or from the card through the communication channel.
I.DATA	JCVM Reference Data: objectref addresses of APDU buffer, JCRE-owned instances of APDU class and byte array for install method.
I.RORD	Remote object reference descriptors which provide information concerning: (i) the identification of the remote object and (ii) the implementation class of the object or the interfaces implemented by the class of the object. The descriptor is the only object’s information to which the CAD can access.

Security attributes linked to these subjects, objects and information are described in

the following table with their values:

Security Attribute	Description
Active Applets	The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels.
Applet Selection Status	"Selected" or "Deselected".
Applet's version number	The version number of an applet (package) indicated in the export file.
Class	Identifies the implementation class of the remote object.
Context	Package AID or "Java Card RE".
Currently Active Context	Package AID or "Java Card RE".
Dependent package AID	Allows the retrieval of the Package AID and Applet's version number ([JCVM22], §4.5.2).
ExportedInfo	Boolean (indicates whether the remote object is exportable or not).
Identifier	The Identifier of a remote object or method is a number that uniquely identifies the remote object or method, respectively.
LC Selection Status	Multiselectable, Non-multiselectable or "None".
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). The owner of a remote object is the applet instance that created the object.
Package AID	The AID of each package indicated in the export file.
Registered Applets	The set of AID of the applet instances registered on the card.
Remote	An object is Remote if it is an instance of a class that directly or indirectly implements the interface java.rmi.Remote.
Resident Packages	The set of AIDs of the packages already loaded on the card.
Returned References	The set of remote object references that have been sent to the CAD during the applet selection session. This attribute is implementation dependent.
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.
AID	Any security domain AID or application/executable load file/executable module AID Stored in the GlobalPlatform Registry
Life Cycle State	Life Cycle State of the card and application Stored in the GlobalPlatform Registry

Security Attribute	Description
Privilege	Privileges for each application including security domain Stored in the GlobalPlatform Registry
Verified	Card content has been verified by associated security domain.
SecureChannel	Secure channel between TOE and CAD has been established.
SecurityLevel	Security level between TOE and CAD, i.e. 0 (none), 1(MAC), or 3 (both encryption and MAC).

(*) 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.GET_ROR(O.APPLET,...)	Retrieves the initial remote object reference of a RMI based applet. This reference is the seed which the CAD client application needs to begin remote method invocations.
OP.INSTANCE_FIELD(O.JAVAOBJECT, field)	Read/Write a field of an instance of a class in the Java programming language.
OP.INVK_VIRTUAL(O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object).
OP.INVK_INTERFACE(O.JAVAOBJECT, method, arg1,...)	Invoke an interface method.
OP.INVOKE(O.RMI_SERVICE,...)	Requests a remote method invocation on the remote object.
OP.JAVA(...)	Any access in the sense of

Operation	Description
	[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.RET_RORD(S.JCRE,S.CAD,I.RORD)	Send a remote object reference descriptor to the CAD.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see [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).
OP.MAC(O.CARD_CONTENT)	Verify card content to be loaded and installed by GP.
OP.DAP(O.CARD_CONTENT)	Authenticate card content to be loaded and installed by GP (DAP Verification).
OP.CCMF(O.CARD_CONTENT, O.REGISTRY)	Perform card content management function: loading and installation, removal, extradition, status transition
OP.SEND(…)	Send a message through the communication channel
OP.RECEIVE(…)	Receive a message through the communication channel

(*) 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.

7.1.1 CoreG_LC security functional requirements

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

7.1.1.1 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:

- **OP.CREATE,**
- **OP.INVK_INTERFACE,**
- **OP.INVK_VIRTUAL,**
- **OP.JAVA,**
- **OP.THROW,**
- **OP.TYPE_ACCESS.**

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

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

FDP_ACF.1/FIREWALL Security attribute based access control

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

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

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

- **R.JAVA.1 ([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".**
- **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.**
- **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.**
- **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.

- **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:

- **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.**
- **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:

- **1) 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.**

[JCSPP] Application note:

FDP_ACF.1.4/FIREWALL:

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

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 can only be 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).
It should be noticed that the invocation of static methods (or access to a static field) is not considered by this policy, as there are no firewall rules. They have no effect on the active context as well and the "acting package" is not the one to which the static method belongs to in this case.
It should be noticed that the Java Card platform, version 2.2.x and version 3 Classic Edition, introduces the possibility for an applet instance to be selected on multiple logical channels at the same time, or accepting other applets belonging to the same package being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same package are either all multiselectable or not ([JCVM22], §2.2.5). Therefore, the selection mode can be regarded as an attribute of packages. No selection mode is defined for a library package.
An applet instance will be considered an active applet instance if it is currently selected in at least one logical channel. An applet instance is the currently selected applet instance only if it is processing the current command. There can only be one currently selected applet instance at a given time. ([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)**.

[JCSPP] Application note:

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

FDP_IFF.1/JCVM Simple security attributes

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

Subject/Information	Security attributes
S.JCVM	**Currently Active Context**

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

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

FDP_IFF.1.3/JCVM The TSF shall enforce the **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**.

[JCSPP] Application note:
The storage of temporary Java Card RE-owned objects references is runtime-enforced ([JCRE22], §6.2.8.1-3).
It should be noticed that this policy essentially applies to the execution of bytecode. Native methods, the Java Card RE itself and possibly some API methods can be granted specific rights or limitations through the FDP_IFF.1.3/JCVM to FDP_IFF.1.5/JCVM elements. The way the Java Card virtual machine manages the transfer of values on the stack and local variables (returned values, uncaught exceptions) from and to internal registers is implementation-dependent. For instance, a returned reference, depending on the implementation of the stack frame, may transit through an internal register prior to being pushed on the stack of the invoker. The returned bytecode would cause more than one OP.PUT operation under this scheme.

FDP_RIP.1/OBJECTS Subset residual information protection

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

[JCSPP] 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**.

[JCSPP] Application note:
The modification of the Selected Applet Context should be 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)**.

[JCSPP] Application note:
The modification of the Currently Active Context should be 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**.

[JCSPP] 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 attributes.
- 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.

[JCSPP] 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. It should be noticed that creation of objects is an operation controlled by the FIREWALL access control SFP. The operation shall fail anyway if the created object would have had security attributes whose value violates FMT_MSA.2.1/FIREWALL_JCVM.

FMT_MSA.3/JCVM Static attribute initialisation

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

FMT_MSA.3.2/JCVM [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/CORE Specification of Management Functions

FMT_SMF.1.1/CORE 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/CORE Security roles

FMT_SMR.1.1/CORE The TSF shall maintain the roles:

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

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

7.1.1.2 APPLICATION PROGRAMMING INTERFACE

The following SFRs are related to the Java Card API.
The whole set of cryptographic algorithms is generally not implemented because of limited memory resources and/or limitations due to exportation. Therefore, the following requirements only apply to the implemented subset.
It should be noticed that the execution of the additional native code is not within the TSF. Nevertheless, access to API native methods from the Java Card System is controlled by TSF because there is no difference between native and interpreted methods in their interface or invocation mechanism.

FCS_CKM.1/CORE Cryptographic key generation

FCS_CKM.1.1/CORE The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **listed in the following table** and specified cryptographic key sizes **listed in the following table** that meet the following: **standards listed in the following table**.

Label	Crypto Algorithm	Crypto Key Sizes	Standards
Protected RSA key generation	**RSA public and private keys computation algorithm, protected against side channel attacks**	**512 up to 2048 bits**	**FIPS PUB 140-2 ISO/IEC 9796-2 PKCS #1 V2.1**
	class KeyPair class KeyBuilder		**[JCAPI222]**
DES/3DES	**class KeyBuilder**	**DES: 56 effective bits (64bits) 3DES 2 keys: 112 effective bits (128bits) 3DES 3 keys: 168 effective bits (192bits)**	**[JCAPI222]**
AES	**class KeyBuilder**	**128, 192 and 256 bits**	**[JCAPI222]**
ECC	**class KeyBuilder**	**112 up to 521 bits**	**[JCAPI222]**
SEED	**class KeyBuilder**	**128 bits**	**[JCAPI222] [FICCS]**

[JCSPP] Application note:

- The keys can be generated and diversified in accordance with [JCAPI22] specification in classes KeyBuilder and KeyPair (at least Session key generation).
- This component shall be instantiated according to the version of the Java Card API applying to the security target and the implemented algorithms ([JCAPI22], [JCAPI221], [JCAPI222] and [JCAPI3]).

[ST] Application note:

- The class KeyPair will be used to generate a RSA key pair.
- The class KeyBuilder will be used to generate storage objects for key values, the storage object will be used by FCS_CKM.2/CORE.

FCS_CKM.2/CORE Cryptographic key distribution

FCS_CKM.2.1/CORE The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **listed in the following table** that meets the following: **standards listed in the following table**.

Label	Crypto Key Distribution Method	Standards
RSA	**setExponent** **setModulus** **setDP1** **setDQ1** **setP** **setPQ** **setQ**	**[JCAPI222]**
DES/3DES	**setKey**	**[JCAPI222]**
AES	**setKey**	**[JCAPI222]**
ECC	**setA** **setB** **setFieldFP** **setG** **setK** **setR** **setS** **setW**	**[JCAPI222]**
SEED	**setKey**	**[JCAPI222]**
	setKey	**[FICCS]**

[JCSPP] Application note:

- Command SetKEY that meets [JCAPI22] specification.
- This component shall be instantiated according to the version of the Java Card API applying to the security target and the implemented algorithms ([JCAPI22], [JCAPI221], [JCAPI222] and [JCAPI3]).

[ST] Application note:

- FCS_CKM.2/CORE will set key values within storage object generated by FCS_CKM.1/CORE.

FCS_CKM.3/CORE Cryptographic key access

FCS_CKM.3.1/CORE The TSF shall perform **access to the keys listed in the following table** in accordance with a specified cryptographic key access method **methods defined in the following table** that meets the following: **standard listed in the following table**.

Label	Crypto Key Access Method	Standards
RSA keys	**methods packages javacard.security and javacardx,crypto**	**[JCAPI222]**
DES/3DES	**methods packages javacard.security and javacardx,crypto**	**[JCAPI222]**
AES	**methods packages javacard.security and javacardx,crypto**	**[JCAPI222]**

ECC	**methods packages javacard.security and javacardx,crypto**	**[JCAPI222]**
SEED	**methods packages javacard.security and javacardx,crypto**	**[JCAPI222]**
	and koreanpackage	**[FICCS]**

[JCSPP] Application note:

- The keys can be accessed as specified in [JCAPI22] Key class.
- This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms ([JCAPI22], [JCAPI221], [JCAPI222] and [JCAPI3]).

FCS_CKM.4/CORE Cryptographic key destruction

FCS_CKM.4.1/CORE The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **listed in the following table** that meets the following: **standard listed in the following table**.

Label	Crypto Key Destruction Method	Standards
RSA keys	**clearKey() method**	**[JCAPI222]**
DES/3DES	**clearKey() method**	**[JCAPI222]**
AES	**clearKey() method**	**[JCAPI222]**
ECC	**clearKey() method**	**[JCAPI222]**
SEED	**clearKey() method**	**[JCAPI222]**
	clearKey() method	**[FICCS]**

[JCSPP] Application note:

- The keys are reset as specified in [JCAPI22] Key class, with the method clearKey(). Any access to a cleared key for ciphering or signing shall throw an exception.
- This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms ([JCAPI22], [JCAPI221], [JCAPI222] and [JCAPI3]).

FCS_COP.1/CORE Cryptographic operation

FCS_COP.1.1/CORE The TSF shall perform **cryptographic operations listed in the following table** in accordance with a specified cryptographic algorithm **listed in the following table** and cryptographic key sizes **listed in the following table** that meet the following: **standards listed in the following table**.

Label	Crypto Operations	Crypto Algorithm	Crypto Key Sizes	Standards
DES / 3DES operation	**encryption, decryption**	**Data Encryption**	**56 effective**	**FIPS PUB 46-3 ISO/IEC 9797-1**

	- in Cipher Block Chaining (CBC) mode - in Electronic Code Book (ECB) mode - in CBC-MAC operating modes	Standard (DES)	bits (64bits)	ISO/IEC 10116
		Triple Data Encryption Standard (3DES)	2 keys: 112 effective bits (128bits) 3 keys: 168 effective bits (192bits)	
RSA operation	RSA recovery (encryption), RSA signature (decryption) without the Chinese Remainder Theorem, RSA signature (decryption) with the Chinese Remainder Theorem	Rivest, Shamir & Adleman's	512 up to 2048 bits	PKCS #1 V2.1
AES operation	cipher operation, inverse cipher operation	Advanced Encryption Standard	128, 192 and 256 bits	FIPS PUB 197
ECDSA operation	general point addition, point expansion, point compression, public scalar multiplication, private scalar multiplication	Elliptic Curves Cryptography on GF(p)	112 up to 521 bits	IEEE 1363-2000, chapter 7 IEEE 1363a-2004
SEED operations	cipher operation, inverse cipher operation		128 bits	ISO/IEC 18033-3, IETF RFC 4269
SHA-1 operation	SHA-1 (secure hash function)	revised Secure Hash Algorithm (SHA-1)	N/A	FIPS PUB 180-1 FIPS PUB 180-2 SO/IEC 10118-3:1998
SHA-256 operation	SHA-256 (secure hash function)	revised Secure Hash	N/A	FIPS PUB 180-1

		Algorithm (SHA-256)		**FIPS PUB 180-2 SO/IEC 10118-3:1998**

[JCSPP] Application note:

- The TOE shall provide a subset of cryptographic operations defined in [JCAPI22] (see javacardx.crypto.Cipher and javacardx.security packages).
- This component shall be instantiated according to the version of the Java Card API applicable to the security target and the implemented algorithms ([JCAPI22], [JCAPI221], [JCAPI222] and [JCAPI3]).

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**.

[JCSPP] Application note:
The events that provoke the de-allocation of a transient object are described in [JCRE22], §5.1.

FDP_RIP.1/APDU Subset residual information protection

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

[JCSPP] 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**.

[JCSPP] 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)**.

[JCSPP] 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 [JCAPI22].

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**.

[JCSPP] 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 ([JCRE22], §4.2.

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 [JCRE22], §7.7, within the bounds of the Commit Capacity ([JCRE22], §7.8), and those described in [JCAPI22]**.

[JCSPP] Application note:

Transactions are a service offered by the APIs to applets. It is also used by some APIs to guarantee the atomicity of some operation. This mechanism is either implemented in Java Card platform or relies on the transaction mechanism offered by the underlying platform.

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

7.1.1.3 CARD SECURITY MANAGEMENT

FAU_ARP.1/CORE Security alarms

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

- **throw an exception,**
- **lock the card session,**
- **reinitialize the Java Card System and its data,**
- **mute the card when exceeding predefined number of secure channel establishment failure,**

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(), [JCAPI22] and ([JCRE22], §7.6.2)**
- **violation of the Firewall or JCVM SFPs,**
- **unavailability of resources,**
- **array overflow,**
- **other runtime errors related to applet's failure (like uncaught exceptions).**

[JCSPP] Application note:

- The developer shall provide the exhaustive list of actual potential security violations the TOE reacts to. For instance, other runtime errors related to applet's failure like uncaught exceptions.
- The bytecode verification defines a large set of rules used to detect a "potential security violation". The actual monitoring of these "events" within the TOE only makes sense when the bytecode verification is performed on-card.
- Depending on the context of use and the required security level, there are cases where the card manager and the TOE must work in cooperation to detect and appropriately react in case of potential security violation. This behavior must be described in this component. It shall detail the nature of the feedback information provided to the card manager (like the identity of the offending application) and the conditions under which the feedback will occur (any occurrence of the java.lang.SecurityException exception).
- The "locking of the card session" may not appear in the policy of the card manager. Such measure should only be taken in case of severe violation detection; the same holds for the re-initialization of the Java Card System. Moreover, the locking should occur when “clean" re-initialization seems to be impossible.
- The locking may be implemented at the level of the Java Card System as a denial of service (through some systematic "fatal error" message or return value) that lasts up to the next "RESET" event, without affecting other components of the card (such as the card manager). Finally, because the installation of applets is a sensitive process, security alerts in this case should also be carefully considered herein.

FDP_SDI.2/CORE Stored data integrity monitoring and action

FDP_SDI.2.1/CORE 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 of D.APP_I_DATA, D.APP_KEYs, D.PIN**.

FDP_SDI.2.2/CORE Upon detection of a data integrity error, the TSF shall **take actions defined in FAU_ARP.1/CORE**.

[JCSPP] Application note:

- Although no such requirement is mandatory in the Java Card specification, at least an exception shall be raised upon integrity errors detection on cryptographic keys, PIN values and their associated security attributes. Even if all the objects cannot be monitored, cryptographic keys and PIN objects shall be considered with particular attention by ST authors as they play a key role in the overall security.
- It is also recommended to monitor integrity errors in the code of the native applications and Java Card applets.
- For integrity sensitive application, their data shall be monitored (D.APP_I_DATA): applications may need to protect information against unexpected modifications, and explicitly control whether a piece of information has been changed between two accesses. For example, maintaining the integrity of an electronic purse's balance is extremely important because this value represents real money. Its modification must be controlled, for illegal ones would denote an important failure of the payment system.
- A dedicated library could be implemented and made available to developers to achieve better security for specific objects, following the same pattern that already exists in cryptographic APIs, for instance.

FPR_UNO.1/CORE Unobservability

FPR_UNO.1.1/CORE The TSF shall ensure that **any user and subject** are unable to observe the operation **End-user (Card Holder) authentication using PIN and cryptographic operations** on **PIN code and cryptographic keys** by **TSF**.

[JCSPP] Application note:

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

FPT_FLS.1/CORE Failure with preservation of secure state

FPT_FLS.1.1/CORE 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/CORE**.

[JCSPP] 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 ([JCRE222]). Behavior of the TOE on power loss and reset is described in [JCRE22], §3.6 and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE222], §3.6.1.

FPT_TDC.1/CORE Inter-TSF basic TSF data consistency

FPT_TDC.1.1/CORE 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/CORE The TSF shall use

- **the rules defined in [JCVM22] specification,**
- **the API tokens defined in the export files of reference implementation,**
- **the rules defined in [VGP]**
- **the rules defined in the [ISO7816], [ISO14443], [EMV42] and [EMVCL201]**

when interpreting the TSF data from another trusted IT product.

[JCSPP] 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.

[ST] Application note:

Therefore, the sharing between the TSF and another trusted IT product includes the sharing between the JCS and Card Manager.

7.1.1.4 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.

[JCSPP] 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/CORE 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: **The Package AID of a package being loaded is not already present within O.REGISTRY**.

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: **none**.

[JCSPP] 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**.

[JCSPP] Application note:

- The installer and the Java Card RE manage other TSF data such as the applet life cycle or CAP files, but this management is implementation specific. Objects in the Java programming language may also try to query AIDs of installed applets through the lookupAID(...) API method.
- The installer, applet deletion manager or even the card manager may be granted the right to modify the list of registered applets' AIDs in specific implementations (possibly needed for installation and deletion; see #.DELETION and #.INSTALL).

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**.

7.1.2 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 PP, 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).

[JCSPP] 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. Implementation-dependent checks may occur on a case-by-case basis to indicate that package files are binary compatible. 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).
- The installation (the invocation of an applet's install method by the installer) is implementation dependent ([JCRE22], §11.2).
- Other rules governing the installation of an applet, that is, its registration to make it SELECTable by giving it a unique AID, are also implementation dependent (see, for example, [JCRE22], §11).

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 [JCRE22] §11.1.4**.

[JCSPP] Application note:
The TOE may provide additional feedback information to the card manager in case of potential security violations (see FAU_ARP.1/CORE).

FPT_RCV.3/Installer Automated recovery without undue loss

FPT_RCV.3.1/Installer When automated recovery from **none** 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 **package/applet load/installation failures**, 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 **0%** for loss of TSF data or objects under the control of the TSF.

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

[JCSPP] 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. Further details on the "maintenance mode" shall be provided in specific implementations. 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 should be 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 may be acceptable to allow any user to restore the system if the TOE enters such a state. However, in practice, this is probably not desirable as the user restoring the system has an opportunity to configure the TOE in such a way as to violate the 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. Precise behavior is left to implementers. This component shall include among the listed failures the deletion of a package/applet. See ([JCRE22], 11.3.4) for possible scenarios. Precise behavior is left to implementers.
- 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 object" and transactional features. See FPT_FLS.1.1/CORE, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT and FDP_ROL.1/FIREWALL.

FPT_RCV.3.3/Installer:

- The quantification is implementation dependent, but some facts can be recalled here. 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 should be limited to the same restrictions of the transaction mechanism.

7.1.3 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. This policy is better thought as a frame to be filled by ST implementers.

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

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

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

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

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

- **R.JAVA.14 ([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 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 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 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 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**.

[JCSPP] 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 orders in which these calls and the dependencies checks are performed are out of the scope of this protection profile.

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**.

[JCSPP] Application note:
Deleted freed resources (both code and data) may be reused, depending on the way they were deleted (logically or physically). Requirements on de-allocation during applet/package deletion are described in [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**.

[JCSPP] Application note:

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

7.1.4 RMIG security functional requirements

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

FDP_ACC.2/JCRMI Complete access control

FDP_ACC.2.1/JCRMI The TSF shall enforce the **JCRMI access control SFP** on **S.CAD, S.JCRE, O.APPLET, O.REMOTE_OBJ, O.REMOTE_MTHD, O.ROR, O.RMI_SERVICE** and all operations among subjects and objects covered by the SFP.

Refinement:

The operations involved in this policy are:

- **OP.GET_ROR,**
- **OP.INVOKE.**

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

FDP_ACF.1/JCRMI Security attribute based access control

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

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

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

- **R.JAVA.18: S.CAD may perform OP.GET_ROR upon O.APPLET only if O.APPLET is the currently selected applet, and there exists an O.RMI_SERVICE with a registered initial reference to an O.REMOTE_OBJ that is owned by O.APPLET.**
- **R.JAVA.19: S.JCRE may perform OP.INVOKE upon O.RMI_SERVICE, O.ROR and O.REMOTE_MTHD only if O.ROR is valid (as defined in [JCRE22], §8.5) and it belongs to the Returned References of O.RMI_SERVICE, and if the Identifier of O.REMOTE_MTHD matches one of the remote methods in the Class of the O.REMOTE_OBJ to which O.ROR makes reference.**

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

FDP_ACF.1.4/JCRMI [Editorially Refined] The TSF shall explicitly deny access of **any subject but S.JCRE to O.REMOTE_OBJ and O.REMOTE_MTHD for the purpose of performing a remote method invocation**.

[JCSPP] Application note:
FDP_ACF.1.2/JCRMI:

- The validity of a remote object reference is specified as a lifetime characterization. The security attributes involved in the rules for determining valid remote object references are the Returned References of the O.RMI_SERVICE and the Active Applets (see FMT_REV.1.1/JCRMI and FMT_REV.1.2/JCRMI). The precise mechanism by which a remote method is invoked on a remote object is defined in detail in ([JCRE22], §8.5.2 and [JCAPI22]).
- Note that the owner of an O.RMI_SERVICE is the applet instance that created the object. The attribute Returned References lists the remote object references that have been sent to the S.CAD during the applet selection session. This attribute is implementation dependent.

FDP_IFC.1/JCRMI Subset information flow control

FDP_IFC.1.1/JCRMI The TSF shall enforce the **JCRMI information flow control SFP** on **S.JCRE, S.CAD, I.RORD and OP.RET_RORD(S.JCRE,S.CAD,I.RORD)**.

[JCSPP] Application note:
FDP_IFC.1.1/JCRMI:

- Array parameters of remote method invocations must be allocated on the card as global arrays objects. References to global arrays cannot be stored in class variables, instance variables or array components. The control of the

flow of that kind of information has already been specified in FDP_IFC.1.1/JCVM.

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

FDP_IFF.1/JCRMI Simple security attributes

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

Subject/Information	Security attributes
I.RORD	**ExportedInfo**

FDP_IFF.1.2/JCRMI The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: **OP.RET_RORD(S.JCRE, S.CAD, I.RORD) is permitted only if the attribute ExportedInfo of I.RORD has the value "true" ([JCRE22], §8.5)**.

FDP_IFF.1.3/JCRMI The TSF shall enforce the **none**.

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

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

[JCSPP] Application note:
The ExportedInfo attribute of I.RORD indicates whether the O.REMOTE_OBJ which I.RORD identifies is exported or not (as indicated by the security attribute ExportedInfo of the O.REMOTE_OBJ).

FMT_MSA.1/EXPORT Management of security attributes

FMT_MSA.1.1/EXPORT The TSF shall enforce the **JCRMI access control SFP** to restrict the ability to **modify** the security attributes: **ExportedInfo of O.REMOTE_OBJ** to **its owner applet**.

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

FMT_MSA.1/REM_REFS Management of security attributes

FMT_MSA.1.1/REM_REFS The TSF shall enforce the **JCRMI access control SFP** to restrict the ability to **modify** the security attributes **Returned References of O.RMI_SERVICE** to **its owner applet**.

FMT_MSA.3/JCRMI Static attribute initialisation

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

FMT_MSA.3.2/JCRMI 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.

[JCSPP] Application note:
FMT_MSA.3.1/JCRMI:

- Remote objects' security attributes are created and initialized at the creation of the object, and except for the ExportedInfo attribute, the values of the attributes are not longer modifiable. The default value of the Exported attribute is true. There is one default value for the Selected Applet Context that is the default applet identifier's context, and one default value for the active context, that is "Java Card RE".

FMT_MSA.3.2/JCRMI:

- The intent is to have none of the identified roles to have privileges with regards to the default values of the security attributes. It should be noticed that creation of objects is an operation controlled by the FIREWALL access control SFP.

FMT_REV.1/JCRMI Revocation

FMT_REV.1.1/JCRMI [Editorially Refined] The TSF shall restrict the ability to revoke **the Returned References of O.RMI_SERVICE to the Java Card RE**.

FMT_REV.1.2/JCRMI The TSF shall enforce the rules t**hat determine the lifetime of remote object references**.

[JCSPP] Application note:
The rules are described in [JCRE22], §8.5.

FMT_SMF.1/JCRMI Specification of Management Functions

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

- **modify the security attribute ExportedInfo of O.REMOTE_OBJ,**
- **modify the security attribute Returned References of O.RMI_SERVICE.**

FMT_SMR.1/JCRMI Security roles

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

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

[JCSPP] Application note:
Applets own remote interface objects and may choose to allow or forbid their exportation, which is managed through a security attribute.

7.1.5 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()**.

[JCSPP] Application note:

- Freed data resources resulting from the invocation of the method javacard.framework.JCSystem.requestObjectDeletion() may be reused. Requirements on de-allocation after the invocation of the method are described in [JCAPI22].
- There is no conflict with FDP_ROL.1 here because of the bounds on the rollback mechanism: the execution of requestObjectDeletion() is not in the scope of the rollback because it must be performed in between APDU command processing, and therefore no transaction can be in progress.

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**.

[JCSPP] Application note:

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

7.1.6 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.

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 **immediate verification of origin**.

[JCSPP] Application note:

FCO_NRO.2.1/CM:

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

FCO_NRO.2.3/CM:

- The exact limitations on the evidence of origin are implementation dependent. In most of the implementations, 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.

Refinement:

The operations involved in the policy are:

- **OP.SEND**
- **OP.RECEIVE**

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.

[JCSPP] Application note:

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

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: **subjects, information and the security attributes listed in the following table.**

Subject/Information	Security attributes
S.INSTALLER	**SecureChannel, SecurityLevel**
S.BCV	**None**
S.CAD	**None**
I.APDU	**SecurityLevel, Verified**

Security attributes	Values
SecureChannel	**Boolean (True or False)**
SecurityLevel	**0 (none)** **1 (MAC)** **3 (Both Encryption and MAC)**
Verified	**Boolean (True or False)**

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:

- **An information flow between S.CAD (on the behalf of S.BCV) and S.INSTALLER is allowed if and only if the following conditions are all satisfied:**
  - **SecureChannel of S.INSTALLER has the value 'True',**
  - **SecurityLevel of I.APDU meets SecurityLevel of S.INSTALLER**
  - **Verified of I.APDU is 'True'.**

FDP_IFF.1.3/CM The TSF shall enforce the **none**.

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

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

[JCSPP] Application note:

FDP_IFF.1.1/CM:

- The security attributes used to enforce the PACKAGE LOADING SFP are implementation dependent. More precisely, they depend on the communication protocol enforced between the CAD and the card. For instance, some of the attributes that can be 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 [GP].

FDP_IFF.1.2/CM:

- The precise set of rules to be enforced by the function is implementation dependent. The whole exchange of messages shall verify at least 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_UIT.1/CM Data exchange integrity

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

FDP_UIT.1.2/CM [Editorially Refined] The TSF shall be able to determine on receipt of user data, whether **modification, deletion, insertion, replay of some of**

the pieces of the application sent by the CAD has occurred.

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

FIA_UID.1/CM Timing of identification

FIA_UID.1.1/CM The TSF shall allow **Get Data, Select Applet, Manage 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.

[JCSPP] Application note:
The list of TSF-mediated actions is implementation-dependent, but package installation requires the user to be identified. Here by user is meant the one(s) that in the Security Target shall be associated to the role(s) defined in the component FMT_SMR.1/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 **query and modify** the security attributes **SecureChannel, SecurityLevel and Verified** to **Card Administrator.**

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 **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: **management of security attributes for PACKAGE LOADING information flow control SFP.**

FMT_SMR.1/CM Security roles

FMT_SMR.1.1/CM The TSF shall maintain the roles **Card Administrator and Application Provider.**

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**.

[JCSPP] Application note:
There is no dynamic package loading on the Java Card platform. New packages can be installed on the card only on demand of the card issuer.

7.1.7 CMGRG security functional requirements

This group contains the security requirements for the card manager. Also, this group contains the security requirements to fulfill GP specific objectives.
The security requirements below help to define a policy for controlling access to card content management operations and for expressing card issuer security concerns. This policy shall be highly dependent on the particular security and card management architecture present in the card. Therefore the policy should be accordingly refined when developing conformant Security Targets.

FDP_ACC.1/CMGR Subset access control

FDP_ACC.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** on **the following list of subjects, objects, and operations.**

- **Subjects: S.OPEN, S.ISD, S.SD**
- **Objects: O.CARD_CONTENT, O.REGISTRY**
- **Operations: OP.MAC(O.CARD_CONTENT), OP.DAP(O.CARD_CONTENT), OP.CCMF(O.CARD_CONTENT, O.REGISTRY)**

FDP_ACF.1/CMGR Security attribute based access control

FDP_ACF.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to objects based on **the following security attributes.**

Subject/Object	Security attributes
S.OPEN	**None**
S.ISD	**AID, Privilege, Life Cycle State**
S.SD	**AID, Privilege, Life Cycle State**
O.CARD_CONTENT	**AID, Verified**
O.REGISTRY	**None**

Security attributes	Values
AID	**Application ID**
Privilege	**Application Privileges defined in [GP]**
Life Cycle State	**card: OP_READY, INITIALIZED, SECURED, CARD_LOCKED, TERMINATED** **security domain: INSTALLED, SELECTABLE, PERSONALIZED, LOCKED**
Verified	**Boolean (True or False)**

[ST] Application note:
Security domain for Application Provider can have either privilege ‘Security Domain Privilege’, ‘Security Domain with DAP Verification Privilege’, or any other application specific privileges. Security Domain for Controlling Authority can have privilege ‘Security Domain with Mandated DAP Verification Privilege’.
ISD inherits the Life Cycle State of the card.

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

- **R.GP.1 Only S.ISD shall be allowed to request S.OPEN to perform OP.CCMF upon O.CARD_CONTENT and O.REGISTRY.**
- **R.GP.2 S.SD shall allow S.ISD to request S.OPEN to perform OP.CCMF upon its own O.CARD_CONTENT and O.REGISTRY.**
- **R.GP.3 S.ISD shall be allowed to perform OP.MAC upon O.CARD_CONTENT.**
- **R.GP.4 If any activated S.SD with Privilege ‘Security Domain with Mandated DAP Verification Privilege’, the S.SD shall perform OP.DAP upon every O.CARD_CONTENT before S.OPEN performs OP.CCMF (for loading and installation).**
- **R.GP.5 If any activated S.SD with Privilege ‘Security Domain with DAP Verification Privilege’, the S.SD shall perform OP.DAP upon its own O.CARD_CONTENT before S.OPEN performs OP.CCMF (for loading and installation).**
- **R.GP.6 S.OPEN shall be allowed to perform OP.CCMF(for loading and installation) upon O.CARD_CONTENT and O.REGISTRY if and only if the following conditions are all satisfied:**
  - **Verified of O.CARD_CONTENT has the value ‘True’,**
  - **OP.DAP on O.CARD_CONTENT is successful (if any activated S.SD mentioned in R.GP.4 and R.GP.5),**
  - **the AID of O.CARD_CONTENT is not already present within O.REGISTRY,**
  - **the associated S.SD’s AID exists within O.REGISTRY and has the relevant Privilege,**
  - **and S.ISD’ or S.SD’s Life Cycle State has valid value.**
- **R.GP.7 S.OPEN shall be allowed to OP.CCMF (for removal except for S.SD’s own removal) upon O.CARD_CONTENT and O.REGISTRY if and only if the following conditions are all satisfied:**
  - **the AID of O.CARD_CONTENT has an entry within O.REGISTRY,**
  - **the AID of O.CARD_CONTENT (including application data) is not currently selected or referenced.**
- **R.GP.8 S.OPEN shall be allowed to OP.CCMF (for extradition) upon O.CARD_CONTENT and O.REGISTRY if and only if the following conditions are all satisfied:**
  - **the AID of O.CARD_CONTENT has an entry within O.REGISTRY,**
  - **the S.SD requesting the extradition is the Security Domain associated with the AID of O.CARD_CONTENT,**
  - **Privilege of the S.SD is Security Domain,**
  - **and S.ISD’ or S.SD’s Life Cycle State has valid value.**

FDP_ACF.1.3/CMGR The TSF shall explicitly authorize access of subjects to objects

based on the following additional rules:

- **R.GP.9 S.ISD and S.SD shall require only the minimum security requirements for GP commands as defined by [VGP].**

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

- **R.GP.10 If Life Cycle State of S.ISD is CARD_LOCKED or TERMINATED, S.ISD shall deny requesting S.OPEN to perform OP.CCMF upon O.CARD_CONTENT and O.REGISTRY.**
- **R.GP.11 If one of the conditions under R.GP.6 fails, S.OPEN shall deny performing OP.CCMF (for loading and installation) upon O.CARD_CONTENT and O.REGISTRY.**
- **R.GP.12 If one of the conditions under R.GP.7 fails, S.OPEN shall deny performing OP.CCMF (for remove) upon O.CARD_CONTENT and O.REGISTRY.**
- **R.GP.13 If one of the conditions under R.GP8 fails, S.OPEN shall deny performing OP.CCMF (for extradition) upon O.CARD_CONTENT and O.REGISTRY.**
- **R.GP.14 If a CCMF process is already in progress on another logical channel, S.OPEN shall deny performing OP.CCMF upon O.CARD_CONTENT and O.REGISTRY.**

FMT_MSA.1/CMGR Management of security attributes

FMT_MSA.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to restrict the ability to **modify, delete, and create** the security attributes **listed in the following table** to **OPEN.**

Abilities	Security attributes
modify, delete, create	**AID, Privilege, and Life Cycle State**
modify	**Verified**

FMT_MSA.3/CMGR Static attribute initialisation

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

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

FMT_SMF.1/CMGR Specification of Management Functions

FMT_SMF.1.1/CMGR The TSF shall be capable of performing the following management functions: **management of security attributes for CARD CONTENT MANAGEMENT access control SFP, which are stored within GlobalPlatform Registry.**

FMT_SMR.1/CMGR Security roles

FMT_SMR.1.1/CMGR The TSF shall maintain the roles: **Card Administrator, Application Provider, and End-user (Card Holder).**

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

FIA_UID.1/CMGR Timing of identification

FIA_UID.1.1/CMGR The TSF shall allow **Get Data, Select Applet, Manage Channel** on behalf of the user to be performed before the user is identified.

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

FIA_AFL.1/GP Authentication failure handling

FIA_AFL.1.1/GP The TSF shall detect when **10 (for the Secure Channel Establishment) and 3 (for the CVM, configurable by Application with CVM privilege)** unsuccessful authentication attempts occur related to **the Secure Channel Establishment and the CVM.**

FIA_AFL.1.2/GP When the defined number of unsuccessful authentication attempts has been **met**, the TSF shall **terminate the card (for Secure Channel Establishment) and block the PIN (for the CVM).**

[ST] Application note:
Basic authentication failure handling applies to any authentication feature managed by the OS or GP.
And to GP when CVM is included.

FIA_ATD.1/GP User attribute definition

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

- **Card Administrator: AID, Life Cycle State, Privilege, ISD Keys**
- **Application Provider: AID, Life Cycle State, Privilege, SD Keys**
- **End-user (Card Holder): AID, Privilege, CVM State, Retry Limit, Retry Counter.**

[ST] Application note:
These are the security attributes associated to GP configuration defined users (e.g. Roles).

FIA_UAU.1/GP Timing of authentication

FIA_UAU.1.1/GP The TSF shall allow **Get Data, Select Applet, Manage Channel** on behalf of the user to be performed before the user is authenticated.

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

[ST] Application note:
Developer should identify in all transactions which operations are allowed before authentication is performed.

FIA_UAU.4/GP Single-use authentication mechanisms

FIA_UAU.4.1/GP The TSF shall prevent reuse of authentication data related to **the Secure Channel Establishment.**

FIA_USB.1/GP User-subject binding

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

FIA_USB.1.2/GP The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **none.**

FIA_USB.1.3/GP The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: **Only privileged Applications shall be allowed to access management services for user security attributes.**

[ST] Application notes:
FIA_USB is related in JCSPP to AID (see 4.1.1.4) but has to be applies also to User security attributes defined in GP configurations

FMT_MOF.1/GP Management of security functions behavior

FMT_MOF.1.1/GP The TSF shall restrict the ability to **disable, enable, modify the behavior of** the functions **listed in the following table** to **Card Administrator.**

Abilities	Functions
enable **modify the behavior of**	**CCMFs**
disable	**transition to previous card life cycle phase**

[ST] Application note:
This SFR applies to OS and GP modification of security function behavior ,by example Loading key for next life-cycle state, disabling previous phase.

FMT_MSA.2/GP Secure security attributes

FMT_MSA.2.1/GP The TSF shall ensure that only secure values are accepted for **Verified and Authenticated.**

[ST] Application note:
Requirement for the TSF to ensure that only secure values Secure are accepted for security attributes applies to JCRE and BCV groups. It shall be applied to GP especially for attributes related to P.LOAD_FILE_VERIFICATION and P.APPLICATION_CODE_VERIFICATION policies

FMT_MTD.1/GP Management of TSF data

FMT_MTD.1.1/GP The TSF shall restrict the ability to **modify, delete, and create** the **GlobalPlatform Registry** to **Card Administrator.**

[ST] Application note:
Application Provider also can perform management of TSF data for its own application by allowing Card Administrator to perform that function.

FMT_SMF.1/GP Specification of Management Functions

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

- **enable and modify the behavior of CCMFs,**
- **disable transition to previous card life cycle phase,**

- **modify, delete, and create the GlobalPlatform Registry.**

Application note:
The SFR should be aligned with FMT of GP

FPT_TST.1/GP TSF testing

FPT_TST.1.1/GP The TSF shall run a suite of self tests **during initial start-up** to demonstrate the correct operation of **the TSF.**

FPT_TST.1.1/GP The TSF shall provide authorised users with the capability to verify the integrity of **parts of TSF data(D.API_DATA, D.CRYPTO, D.JCS_CODE, D.JCS_DATA, D.SEC_DATA, D.OS_DATA, D.OS_CODE, D.ISD_KEYS, D.SD_KEYS).**

FPT_TST.1.3/GP The TSF shall provide authorised users with the capability to verify the integrity of **parts of TSF(Card manager, GP API, JCRE, JCVM, JCAPI)**

7.1.8 SCPG security functional requirements

This group contains the security requirements for the smart card platform, that is, operating system and chip that the Java Card System is implemented upon. These requirements apply to the GP platform.

FCS_RNG.1/SCP Random number generation

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

FCS_RNG.1.2/SCP The TSF shall provide random numbers that meet **P2 class of BSI-AIS31**.

[ST] Application note:
This requirement comes from the [ICST].

FPT_FLS.1/SCP Failure with preservation of secure state

FPT_FLS.1.1/SCP The TSF shall preserve a secure state when the following types of failures occur: **exposure to operating conditions which may not be tolerated according to the requirement FRU_FLT.1 and where therefore a malfunction could occur.**

[ST] Application note:
This requirement comes from FPT_FLS.1 in the [ICST].

FPT_PHP.3/SCP Resistance to physical attack

FPT_PHP.3.1/SCP The TSF shall resist **physical manipulation and physical probing** to the **TSF** by responding automatically such that the TSP is not violated.

[ST] Application note:
This requirement comes from the [ICST].

FPT_RCV.3/SCP Automated recovery without undue loss

FPT_RCV.3.1/SCP When automated recovery from **none** is not possible, the TSF shall enter a maintenance mode where the ability to return the TOE to a secure state

is provided.

FPT_RCV.3.2/SCP For **all cases**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/SCP The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **0%** for loss of TSF data or objects within the TSC.

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

FPT_RCV.4/SCP Function recovery

FPT_RCV.4.1/SCP The TSF shall ensure that **reading from and writing to static and objects' fields interrupted by power loss** have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

[ST] Application note:
This requirement comes from the specification of the Java Card platform but is obviously supported in the implementation by a low-level mechanism of the smart Card.

FRU_FLT.1/SCP Degraded fault tolerance

FRU_FLT.1.1/SCP The TSF shall ensure the operation of **JCS and CM capabilities** when the following failures occur: **lack of EEPROM, random number generator failure, crypto co-processor failure, RAM read/write failure, card tearing, and power failure**.

[ST] Application note :
This requirement shall be used to specify the list of SCP capabilities supporting the Java Card System/CM that will still be operational at the occurrence of the mentioned failures (EEPROM worn out, lack of EEPROM, random generator failure). The minimum can be the function that allows to reset /mute /block the card in case of failure

7.1.9 Compatibility statement of SFRs

SFRs for the TOE in [ICST] are all based on the PP [PP0035], and the [ICST] specifies some additional SFRs. The relevant SFRs for the TOE of the [ICST] are not contradictory to those of the composite ST.

[ICST]	composite ST
FRU_FLT.2	FRU_FLT.1/SCP
FPT_FLS.1	FPT_FLS.1/SCP
FMT_LIM.1	not applicable
FMT_LIM.2	not applicable
FAU_SAS.1	not applicable
FPT_PHP.3	FPT_PHP.3/SCP
FDP_ITT.1	FPR_UNO.1/CORE
FPT_ITT.1	FPR_UNO.1/CORE

[ICST]	composite ST
FDP_IFC.1	FPR_UNO.1/CORE
FCS_RNG.1	FCS_RNG.1/SCP
FCS_COP.1	FCS_COP.1/CORE
FCS_CKM.1	FCS_CKM.1/CORE
FDP_ACC.2	FDP_ACC.2/FIREWALL FDP_ACC.2/ADEL FDP_ACC.2/JCRMI FDP_ACC.1/CMGR
FDP_ACF.1	FDP_ACF.1/FIREWALL FDP_ACF.1/ADEL FDP_ACF.1/JCRMI FDP_ACF.1/CMGR
FMT_MSA.3	FMT_MSA.3/FIREWALL FMT_MSA.3/ADEL FMT_MSA.3/JCRMI FMT_MSA.3/CMGR
FMT_MSA.1	FMT_MSA.1/JCRE FMT_MSA.1/ADEL FMT_MSA.1/EXPORT FMT_MSA.1/REM_REFS FMT_MSA.1/CMGR

7.2 Security assurance requirements

The security assurance requirement level is EAL4 augmented with ALC_DVS.2 and AVA_VAN.5.

7.2.1 Compatibility statement of SARs

The composite ST claims conformance to the same assurance package included in the [ICST], therefore the security assurance requirements of the composite ST represent a subset of those of the [ICST].

7.3 Security requirements rationale

7.3.1 Security objectives for the TOE

7.3.1.1 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/REM_REFS, FMT_MSA.1/EXPORT, FMT_MSA.1/ADEL, FMT_MSA.1/CM, FMT_MSA.3/JCRMI, FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.3/CM, FMT_SMF.1/CM, FMT_SMF.1/ADEL, FMT_SMF.1/ADEL, FMT_SMF.1/JCRMI, 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).

7.3.1.2 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), the JCRMI access control policy (FDP_ACC.2/JCRMI, FDP_ACF.1/JCRMI) and the functional requirement FDP_ITC.2/Installer. The functional requirements of the class FMT (FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_SMR.1/Installer, FMT_SMR.1/CORE, FMT_SMF.1/CORE, FMT_SMR.1/ADEL, FMT_SMR.1/JCRMI, FMT_SMF.1/ADEL, FMT_SMF.1/JCRMI, 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/EXPORT, FMT_MSA.1/REM_REFS, FMT_MSA.3/JCRMI, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_REV.1/JCRMI) 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, 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/CORE), 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/CORE, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FAU_ARP.1/CORE). Its security-critical parts and procedures are also protected: safe recovery from failure is ensured

(FPT_RCV.3/Installer), applets' installation may be cleanly aborted (FDP_ROL.1/FIREWALL), communication with external users and their internal subjects is well-controlled (FDP_ITC.2/Installer, FIA_ATD.1/AID, FIA_USB.1/AID) to prevent alteration of TSF data (also protected by components of the FPT class).
Almost every objective and/or functional requirement indirectly contributes to this one too.

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/CORE, FPT_FLS.1/ADEL, FPT_FLS.1/CORE, 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/CORE, FMT_SMF.1/CORE, FMT_SMR.1/ADEL, FMT_SMR.1/JCRMI, FMT_SMF.1/ADEL, FMT_SMF.1/JCRMI, FMT_SMF.1/CM and FMT_SMR.1/CM).

7.3.1.3 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/CORE which defines TSF reaction upon detection of a potential security violation.

O.CIPHER This security objective is directly covered by FCS_CKM.1/CORE, FCS_CKM.2/CORE, FCS_CKM.3/CORE, FCS_CKM.4/CORE and FCS_COP.1/CORE. The SFR FPR_UNO.1/CORE 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/CORE as well. Precisely it is met by the following components: FCS_CKM.1/CORE, FCS_CKM.2/CORE, FCS_CKM.3/CORE, FCS_CKM.4/CORE, FCS_COP.1/CORE, FPR_UNO.1/CORE, 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/CORE, FDP_ROL.1/FIREWALL and FDP_SDI.2/CORE 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.REMOTE The access to the TOE's internal data and the flow of information from the card to the CAD required by the JCRMI service is under control of the JCRMI access control policy (FDP_ACC.2/JCRMI, FDP_ACF.1/JCRMI) and the JCRMI information flow control policy (FDP_IFC.1/JCRMI, FDP_IFF.1/JCRMI). The security functional requirements of the class FMT (FMT_MSA.1/EXPORT, FMT_MSA.1/REM_REFS, FMT_MSA.3/JCRMI, FMT_REV.1/JCRMI and FMT_SMR.1/JCRMI) included in the group RMIG also contribute to meet this objective.

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).

7.3.1.4 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.

7.3.1.5 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).

7.3.1.6 Reassignment

O.CARD-MANAGEMENT This security objective specifies card management functions, and the security functional requirements FDP_ACC.1/CMGR, FDP_ACF.1/CMGR, FMT_MSA.1/CMGR, FMT_MSA.3/CMGR, FMT_SMF.1/CMGR, FMT_SMR.1/CMGR, FIA_UID.1/CMGR in CMGRG contribute to meet this objective.

O.SCP.IC This security objective specifies IC security features against physical attacks, and the security functional requirement FPT_PHP.3/SCP and FPR_UNO.1/CORE in SCPG contributes to meet this objective.

O.SCP.RECOVERY This security objective specifies recovery function after abnormal situation, and the security functional requirements FPT_FLS.1/SCP, FPT_RCV.3/SCP, FRU_FLT.1/SCP in SCPG contribute to meet this objective.

O.SCP.SUPPORT This security objective specifies IC platform's supportive functions for TOE operation, and the security functional requirements FPT_RCV.3/SCP, FPT_RCV.4/SCP, FCS_RNG.1/SCP in SCPG contributes to meet this objective.

7.3.1.7 Additional security objectives for the TOE

O.PROTECT_DATA This security objective specifies protection of sensitive information stored in memories. This security objective is satisfied by the following SFRs: FCS_CKM.1/CORE, FCS_CKM.2/CORE, FCS_CKM.3/CORE, FCS_CKM.4/CORE, FCS_COP.1/CORE, FDP_ACC.1/CMGR, FDP_ACF.1/CMGR, FDP_RIP.1/OBJECTS, FDP_RIP.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/TRANSIENT, FDP_SDI.2/CORE, FIA_AFL.1/GP, FIA_ATD.1/GP, FIA_UAU.1/GP, FIA_UID.1/CMGR, FIA_USB.1/GP, FPR_UNO.1/CORE, which impose access control to sensitive information through cryptography, user identification and authentication, access control policy, the cleared contents of the re-allocated block, and the unobservability of operations on sensitive information.

O.OS_OPERATE This security objective specifies continued correct operation of security functions. This security objective is directly satisfied by the following SFRs: FDP_SDI.2/CORE, FIA_AFL.1/GP, FIA_ATD.1/GP, FMT_MSA.2/GP, FPT_FLS.1/SCP, FPT_RCV.3/SCP, FPT_RCV.4/SCP, FPT_TST.1/GP, FRU_FLT.1/SCP, which imposes integrity check, secure security attributes, and secure smart card platform.

O.SIDE_CHANNEL This security objective specifies protection against disclosure of confidential data stored and/or processed in the smart card IC. This security objective is directly satisfied by FPR_UNO.1/CORE.

O.FAULT_PROTECT This security objective specifies protection against incorrect operation due to environmental conditions. This security objective is satisfied by FPT_FLS.1/SCP, FPT_PHP.3/SCP, which impose physical protection of the TOE.

O.RND This security objective is directly satisfied by FCS_RND.1/SCP.

O.ROLES This security objective is satisfied by the following SFRs: FIA_AFL.1/GP,

FIA_ATD.1/GP, FIA_USB.1/GP, FMT_SMR.1/CMGR, which imposes roles to be recognized in the TOE.

O.CARD_ADMIN This security objective is satisfied by the following SFRs: FIA_ATD.1/GP, FIA_USB.1/GP, FMT_MOF.1/GP, FMT_MSA.1/CMGR, FMT_MSA.3/CMGR, FMT_MTD.1/GP, FMT_SMF.1/GP, which imposes the Card Administrator with means to perform secure CCMFs.

O.APPLICATION_PROVIDER_PRE-APPROVAL This security objective is satisfied by the following SFRs: FIA_ATD.1/GP, FMT_MOF.1/GP, FMT_MSA.1/CMGR, FMT_SMF.1/GP, which imposes Application Provider to allow the Card Administrator to perform CCMFs.

O.LOAD_FILE_VERIFICATION This security objective is satisfied by the following SFRs: FMT_MSA.1/CMGR, FMT_MSA.2/GP, FMT_MSA.3/CMGR, FMT_SMR.1/CMGR, which imposes load file verification function.

O.APPLICATION_CODE_VERIFICATION This security objective is satisfied by the following SFRs: FDP_ITC.1/Installer, FMT_MSA.1/CMGR, FMT_MSA.2/GP, FMT_MSA.3/CMGR, FMT_SMR.1/CMGR, which imposes means to verify that the byte code and other forms of application code verification has been performed.

O.SECURE_COMM This security objective is satisfied by the following SFRs: FCS_CKM.3/CORE, FCS_COP.1/CORE, FIA_UAU.4/GP, which imposes secure channel.

O.CARDHOLDER_VERIFICATION This security objective is satisfied FIA_AFL.1/GP, FIA_ATD.1/GP, FIA_USB.1/GP, which imposes user authentication.

7.3.2 Rationale tables of security objectives for the TOE and SFRs

Security Objectives for the TOE	SFRs	Rationale
O.SID	FIA_ATD.1/AID, FIA_UID.2/AID, FMT_MSA.1/JCRE, FMT_MSA.3/JCRMI, FMT_MSA.1/REM_REFS, FMT_MSA.1/EXPORT, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.1/CM, FMT_MSA.3/CM, FDP_ITC.2/Installer, FMT_SMF.1/CM, FMT_SMF.1/ADEL, FMT_SMF.1/JCRMI, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FIA_USB.1/AID, FMT_MSA.1/JCVM, FMT_MSA.3/JCVM	Section 7.3.1.1
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,	Section 7.3.1.2

Security Objectives for the TOE	SFRs	Rationale
	FMT_MSA.3/FIREWALL, FMT_SMR.1, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FMT_MSA.1/EXPORT, FMT_MSA.1/REM_REFS, FMT_MSA.3/JCRMI, FMT_REV.1/JCRMI, FMT_SMR.1/JCRMI, FMT_MSA.1/JCRE, FDP_ITC.2/Installer, FDP_ACC.2/JCRMI, FDP_ACF.1/JCRMI, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FMT_SMF.1/ADEL, FMT_SMF.1/JCRMI, 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	
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 7.3.1.2
O.GLOBAL_ARRAYS_INTEG	DP_IFC.1/JCVM, FDP_IFF.1/JCVM	Section 7.3.1.2
O.NATIVE	FDP_ACF.1/FIREWALL	Section 7.3.1.2
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	Section 7.3.1.2
O.REALLOCATION	FDP_RIP.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS	Section 7.3.1.2
O.RESOURCES	FAU_ARP.1, FDP_ROL.1/FIREWALL, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMR.1/ADEL, FMT_SMR.1/JCRMI, 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/JCRMI, FMT_SMF.1/CM, FMT_SMF.1, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE	Section 7.3.1.2
O.ALARM	FPT_FLS.1/Installer, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL, FAU_ARP.1	Section 7.3.1.3
O.CIPHER	FCS_CKM.1, FCS_CKM.2, FCS_CKM.3,	Section

Security Objectives for the TOE	SFRs	Rationale
	FCS_CKM.4, FCS_COP.1, FPR_UNO.1	7.3.1.3
O.KEY-MNGT	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_SDI.2, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT	Section 7.3.1.3
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 7.3.1.3
O.REMOTE	FDP_ACC.2/JCRMI, FDP_ACF.1/JCRMI, FDP_IFC.1/JCRMI, FDP_IFF.1/JCRMI, FMT_MSA.1/EXPORT, FMT_MSA.1/REM_REFS, FMT_MSA.3/JCRMI, FMT_REV.1/JCRMI, FMT_SMR.1/JCRMI	Section 7.3.1.3
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 7.3.1.3
O.OBJ-DELETION	FDP_RIP.1/ODEL, FPT_FLS.1/ODEL	Section 7.3.1.4
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 7.3.1.5
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 7.3.1.5
O.INSTALL	FDP_ITC.2/Installer, FPT_RCV.3/Installer, FPT_FLS.1/Installer	Section 7.3.1.5
O.CARD-MANAGEMENT	FDP_ACC.1/CMGR, FDP_ACF.1/CMGR, FMT_MSA.1/CMGR, FMT_MSA.3/CMGR, FMT_SMF.1/CMGR, FMT_SMR.1/CMGR, FIA_UID.1/CMGR	Section 7.3.1.6
O.SCP.IC	FPT_PHP.3/SCP, FPR_UNO.1/CORE	Section 7.3.1.6
O.SCP.RECOVERY	FPT_FLS.1/SCP, FPT_RCV.3/SCP, FRU_FLT.1/SCP	Section 7.3.1.6
O.SCP.SUPPORT	FPT_RCV.3/SCP, FPT_RCV.4/SCP,	Section

Security Objectives for the TOE	SFRs	Rationale
	FCS_RNG.1/SCP	7.3.1.6
O.PROTECT_DATA	FCS_CKM.1/CORE, FCS_CKM.2/CORE, FCS_CKM.3/CORE, FCS_CKM.4/CORE, FCS_COP.1/CORE, FDP_ACC.1/CMGR, FDP_ACF.1/CMGR, FDP_RIP.1/OBJECTS, FDP_RIP.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/TRANSIENT, FDP_SDI.2/CORE, FIA_AFL.1/GP, FIA_ATD.1/GP, FIA_UAU.1/GP, FIA_UID.1/CMGR, FIA_USB.1/GP, FPR_UNO.1/CORE	Section 7.3.1.7
O.OS_OPERATE	FDP_SDI.2/CORE, FIA_AFL.1/GP, FIA_ATD.1/GP, FMT_MSA.2/GP, FPT_FLS.1/SCP, FPT_RCV.3/SCP, FPT_RCV.4/SCP, FPT_TST.1/GP, FRU_FLT.1/SCP	Section 7.3.1.7
O.SIDE_CHANNEL	FPR_UNO.1/CORE	Section 7.3.1.7
O.FAULT_PROTECT	FPT_FLS.1/SCP, FPT_PHP.3/SCP	Section 7.3.1.7
O.RND	FCS_RND.1/SCP	Section 7.3.1.7
O.ROLES	FIA_AFL.1/GP, FIA_ATD.1/GP, FIA_USB.1/GP, FMT_SMR.1/CMGR	Section 7.3.1.7
O.CARD_ADMIN	FIA_ATD.1/GP, FIA_USB.1/GP, FMT_MOF.1/GP, FMT_MSA.1/CMGR, FMT_MSA.3/CMGR, FMT_MTD.1/GP, FMT_SMF.1/GP	Section 7.3.1.7
O.APPLICATION_PROVIDER_PRE-APPROVAL	FIA_ATD.1/GP, FMT_MOF.1/GP, FMT_MSA.1/CMGR, FMT_SMF.1/GP	Section 7.3.1.7
O.LOAD_FILE_VERIFICATION	FMT_MSA.1/CMGR, FMT_MSA.2/GP, FMT_MSA.3/CMGR, FMT_SMR.1/CMGR	Section 7.3.1.7
O.APPLICATION_CODE_VERIFICATION	FDP_ITC.2/Installer, FMT_MSA.1/CMGR, FMT_MSA.2/GP, FMT_MSA.3/CMGR, FMT_SMR.1/CMGR	Section 7.3.1.7
O.SECURE_COMM	FCS_CKM.3/CORE, FCS_COP.1/CORE, FIA_UAU.4/GP	Section 7.3.1.7
O.CARDHOLDER_VERIFICATION	FIA_AFL.1/GP, FIA_ATD.1/GP, FIA_USB.1/GP	Section 7.3.1.7

Table9. Security Objectives for the TOE and SFRs - Coverage

	SFRs	Security Objectives for the TOE
CoreG_LC	FDP_ACC.2/FIREWALL	O.FIREWALL, O.OPERATE, O.PIN-MNGT
	FDP_ACF.1/FIREWALL	O.FIREWALL, O.NATIVE,

	SFRs	Security Objectives for the TOE
		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, O.PROTECT_DATA
	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/CORE	O.FIREWALL, O.RESOURCES
	FMT_SMR.1/CORE	O.FIREWALL, O.RESOURCES
	FCS_CKM.1/CORE	O.CIPHER, O.KEY-MNGT, O.PROTECT_DATA
	FCS_CKM.2/CORE	O.CIPHER, O.KEY-MNGT, O.PROTECT_DATA
	FCS_CKM.3/CORE	O.CIPHER, O.KEY-MNGT, O.PROTECT_DATA, O.SECURE_COMM
	FCS_CKM.4/CORE	O.CIPHER, O.KEY-MNGT, O.PROTECT_DATA
	FCS_COP.1/CORE	O.CIPHER, O.KEY-MNGT, O.PROTECT_DATA, O.SECURE_COMM
	FDP_RIP.1/ABORT	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION, O.PROTECT_DATA
	FDP_RIP.1/APDU	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION, O.PROTECT_DATA
	FDP_RIP.1/bArray	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION, O.PROTECT_DATA

	SFRs	Security Objectives for the TOE
	FDP_RIP.1/KEYS	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION, O.PROTECT_DATA
	FDP_RIP.1/TRANSIENT	O.GLOBAL_ARRAYS_CONFID, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.REALLOCATION, O.PROTECT_DATA
	FDP_ROL.1/FIREWALL	O.OPERATE, O.RESOURCES, O.PIN-MNGT, O.TRANSACTION
	FAU_ARP.1/CORE	O.OPERATE, O.RESOURCES, O.ALARM
	FDP_SDI.2/CORE	O.KEY-MNGT, O.PIN-MNGT, O.PROTECT_DATA, O.OS_OPERATE
	FPR_UNO.1/CORE	O.CIPHER, O.KEY-MNGT, O.PIN-MNGT, O.SCP.IC, O.PROTECT_DATA, O.SIDE_CHANNEL
	FPT_FLS.1/CORE	O.OPERATE, O.RESOURCES, O.ALARM
	FPT_TDC.1/CORE	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
InstG	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
ADELG	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,

	SFRs	Security Objectives for the TOE
		O.RESOURCES
	FMT_SMR.1/ADEL	O.FIREWALL, O.RESOURCES, O.DELETION
	FPT_FLS.1/ADEL	O.OPERATE, O.RESOURCES, O.ALARM, O.DELETION
RMIG	FDP_ACC.2/JCRMI	O.FIREWALL, O.REMOTE
	FDP_ACF.1/JCRMI	O.FIREWALL, O.REMOTE
	FDP_IFC.1/JCRMI	O.REMOTE
	FDP_IFF.1/JCRMI	O.REMOTE
	FMT_MSA.1/EXPORT	O.SID, O.FIREWALL, O.REMOTE
	FMT_MSA.1/REM_REFS	O.SID, O.FIREWALL, O.REMOTE
	FMT_MSA.3/JCRMI	O.SID, O.FIREWALL, O.REMOTE
	FMT_REV.1/JCRMI	O.FIREWALL, O.REMOTE
	FMT_SMF.1/JCRMI	O.SID, O.FIREWALL, O.RESOURCES
	FMT_SMR.1/JCRMI	O.FIREWALL, O.RESOURCES, O.REMOTE
ODELG	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
CarG	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
CMGRG	FDP_ACC.1/CMGR	O.CARD-MANAGEMENT, O.PROTECT_DATA
	FDP_ACF.1/CMGR	O.CARD-MANAGEMENT, O.PROTECT_DATA
	FMT_MSA.1/CMGR	O.CARD-MANAGEMENT, O.CARD_ADMIN, O.APPLICATION_PROVIDER_PRE-APPROVAL, O.LOAD_FILE_VERIFICATION, O.APPLICATION_CODE_VERIFICATION
	FMT_MSA.3/CMGR	O.CARD-MANAGEMENT, O.CARD_ADMIN,

	SFRs	Security Objectives for the TOE
		O.LOAD_FILE_VERIFICATION, O.APPLICATION_CODE_VERIFICATION
	FMT_SMF.1/CMGR	O.CARD-MANAGEMENT
	FMT_SMR.1/CMGR	O.CARD-MANAGEMENT, O.ROLES, O.LOAD_FILE_VERIFICATION, O.APPLICATION_CODE_VERIFICATION
	FIA_UID.1/CMGR	O.CARD-MANAGEMENT, O.PROTECT_DATA
	FIA_AFL.1/GP	O.PROTECT_DATA, O.OS_OPERATE, O.ROLES, O.CARDHOLDER_VERIFICATION
	FIA_ATD.1/GP	O.PROTECT_DATA, O.OS_OPERATE, O.ROLES, O.CARD_ADMIN, O.APPLICATION_PROVIDER_PRE-APPROVAL, O.CARDHOLDER_VERIFICATION
	FIA_UAU.1/GP	O.PROTECT_DATA
	FIA_UAU.4/GP	O.SECURE_COMM
	FIA_USB.1/GP	O.PROTECT_DATA, O.ROLES, O.CARD_ADMIN, O.CARDHOLDER_VERIFICATION
	FMT_MOF.1/GP	O.CARD_ADMIN, O.APPLICATION_PROVIDER_PRE-APPROVAL
	FMT_MSA.2/GP	O.OS_OPERATE, O.LOAD_FILE_VERIFICATION, O.APPLICATION_CODE_VERIFICATION
	FMT_MTD.1/GP	O.CARD_ADMIN
	FMT_SMF.1/GP	O.CARD_ADMIN, O.APPLICATION_PROVIDER_PRE-APPROVAL
	FPT_TST.1/GP	O.OS_OPERATE
SCPG	FCS_RNG.1/SCP	O.SCP.SUPPORT, O.RND
	FPT_FLS.1/SCP	O.SCP.RECOVERY, O.OS_OPERATE, O.FAULT_PROTECT
	FPT_PHP.3/SCP	O.SCP.IC, O.FAULT_PROTECT
	FPT_RCV.3/SCP	O.SCP.RECOVERY, O.SCP.SUPPORT, O.OS_OPERATE
	FPT_RCV.4/SCP	O.SCP.SUPPORT, O.OS_OPERATE
	FRU_FLT.1/SCP	O.SCP.RECOVERY, O.OS_OPERATE

Table10. SFRs and Security Objectives for the TOE

7.3.3 Dependencies

7.3.3.1 SFRs dependencies

SFRs	CC Dependencies	Satisfied Dependencies
FDP_ACC.2/FIREWALL	FDP_ACF.1	FDP_ACF.1/FIREWALL
FDP_ACF.1/FIREWALL	FDP_ACC.1 FMT_MSA.3	FDP_ACC.2/FIREWALL FMT_MSA.3/FIREWALL
FDP_IFC.1/JCVM	FDP_IFF.1	FDP_IFF.1/JCVM
FDP_IFF.1/JCVM	FDP_IFC.1 FMT_MSA.3	FDP_IFC.1/JCVM FMT_MSA.3/JCVM
FDP_RIP.1/OBJECTS	No dependencies	
FMT_MSA.1/JCRE	[FDP_ACC.1 or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	FDP_ACC.2/FIREWALL FMT_SMR.1/CORE FMT_SMF.1/CORE
FMT_MSA.1/JCVM	[FDP_ACC.1 or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	FDP_ACC.2/FIREWALL FDP_IFC.1/JCVM FMT_SMR.1/CORE FMT_SMF.1/CORE
FMT_MSA.2/FIREWALL_JCVM	[FDP_ACC.1 or FDP_IFC.1] FMT_MSA.1 FMT_SMR.1	FDP_ACC.2/FIREWALL FDP_IFC.1/JCVM FMT_MSA.1/JCRE FMT_MSA.1/JCVM FMT_SMR.1
FMT_MSA.3/FIREWALL	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1/JCRE FMT_MSA.1/JCVM FMT_SMR.1
FMT_MSA.3/JCVM	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1/JCVM FMT_SMR.1
FMT_SMF.1/CORE	No dependencies	
FMT_SMR.1/CORE	FIA_UID.1	FIA_UID.2/AID
FCS_CKM.1/CORE	[FCS_CKM.2 or FCS_COP.1] FCS_CKM.4	FCS_CKM.2/CORE FCS_CKM.4/CORE
FCS_CKM.2/CORE	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	FCS_CKM.1/CORE FCS_CKM.4/CORE
FCS_CKM.3/CORE	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	FCS_CKM.1/CORE FCS_CKM.4/CORE
FCS_CKM.4/CORE	[FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1/CORE
FCS_COP.1/CORE	[FDP_ITC.1 or	

SFRs	CC Dependencies	Satisfied Dependencies
	FDP_ITC.2 or FCS_CKM.1] FCS_CKM.4	 FCS_CKM.1/CORE FCS_CKM.4/CORE
FDP_RIP.1/ABORT	No dependencies	
FDP_RIP.1/APDU	No dependencies	
FDP_RIP.1/bArray	No dependencies	
FDP_RIP.1/KEYS	No dependencies	
FDP_RIP.1/TRANSIENT	No dependencies	
FDP_ROL.1/FIREWALL	[FDP_ACC.1 or FDP_IFC.1]	FDP_ACC.2/FIREWALL FDP_IFC.1/JCVM
FAU_ARP.1/CORE	FAU_SAA.1	Not satisfied
FDP_SDI.2/CORE	No dependencies	
FPR_UNO.1/CORE	No dependencies	
FPT_FLS.1/CORE	No dependencies	
FPT_TDC.1/CORE	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_SMR.1 FMT_SMF.1	FMT_SMR.1/CORE FMT_SMF.1/CORE
FMT_MTD.3/JCRE	FMT_MTD.1	FMT_MTD.1/JCRE
FDP_ITC.2/Installer	[FDP_ACC.1 or FDP_IFC.1] [FTP_ITC.1 or FTP_TRP.1] FPT_TDC.1	 FDP_IFC.2/CM FTP_ITC.1/CM FPT_TDC.1
FMT_SMR.1/Installer	FIA_UID.1	Not satisfied
FPT_FLS.1/Installer	No dependencies	
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 FMT_MSA.3	FDP_ACC.2/ADEL, FMT_MSA.3/ADEL
FDP_RIP.1/ADEL	No dependencies	
FMT_MSA.1/ADEL	[FDP_ACC.1 or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	FDP_ACC.2/ADEL FMT_SMR.1/ADEL FMT_SMF.1/ADEL
FMT_MSA.3/ADEL	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1/ADEL FMT_SMR.1/ADEL
FMT_SMF.1/ADEL	No dependencies	
FMT_SMR.1/ADEL	FIA_UID.1	Not satisfied
FPT_FLS.1/ADEL	No dependencies	
FDP_ACC.2/JCRMI	FDP_ACF.1	FDP_ACF.1/JCRMI
FDP_ACF.1/JCRMI	FDP_ACC.1 FMT_MSA.3	FDP_ACC.2/JCRMI FMT_MSA.3/JCRMI
FDP_IFC.1/JCRMI	FDP_IFF.1	FDP_IFF.1/JCRMI
FDP_IFF.1/JCRMI	FDP_IFC.1	FDP_IFC.1/JCRMI,

SFRs	CC Dependencies	Satisfied Dependencies
	FMT_MSA.3	FMT_MSA.3/JCRMI
FMT_MSA.1/EXPORT	[FDP_ACC.1 or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	FDP_ACC.2/JCRMI FMT_SMR.1/JCRMI FMT_SMF.1/JCRMI
FMT_MSA.1/REM_REFS	[FDP_ACC.1 or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	FDP_ACC.2/JCRMI FMT_SMR.1/JCRMI FMT_SMF.1/JCRMI
FMT_MSA.3/JCRMI	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1/EXPORT FMT_MSA.1/REM_REFS FMT_SMR.1/JCRMI
FMT_REV.1/JCRMI	FMT_SMR.1	FMT_SMR.1/JCRMI
FMT_SMF.1/JCRMI	No dependencies	
FMT_SMR.1/JCRMI	FIA_UID.1	FIA_UID.2/AID
FDP_RIP.1/ODEL	No dependencies	
FPT_FLS.1/ODEL	No dependencies	
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 FMT_MSA.3	FDP_IFC.2/CM FMT_MSA.3/CM
FDP_UIT.1/CM	[FDP_ACC.1 or FDP_IFC.1] [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.1] FMT_SMR.1 FMT_SMF.1	 FDP_IFC.2/CM FMT_SMR.1/CM FMT_SMF.1/CM
FMT_MSA.3/CM	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1/CM FMT_SMR.1/CM
FMT_SMF.1/CM	No dependencies	
FMT_SMR.1/CM	FIA_UID.1	FIA_UID.1/CM
FTP_ITC.1/CM	No dependencies	
FDP_ACC.1/CMGR	FDP_ACF.1	FDP_ACF.1/CMGR
FDP_ACF.1/CMGR	FDP_ACC.1 FMT_MSA.3	FDP_ACC.1/CMGR FMT_MSA.3/CMGR
FMT_MSA.1/CMGR	[FDP_ACC.1 or FDP_IFC.1] FMT_SMR.1 FMT_SMF.1	FDP_ACC.1/CMGR FMT_SMR.1/CMGR FMT_SMF.1/CMGR
FMT_MSA.3/CMGR	FMT_MSA.1 FMT_SMR.1	FMT_MSA.1/CMGR FMT_SMR.1/CMGR
FMT_SMF.1/CMGR	No dependencies	
FMT_SMR.1/CMGR	FIA_UID.1	FIA_UID.1/CMGR
FIA_UID.1/CMGR	No dependencies	

SFRs	CC Dependencies	Satisfied Dependencies
FIA_AFL.1/GP	FIA_UAU.1	FIA_UAU.1/GP
FIA_ATD.1/GP	No dependencies	
FIA_UAU.1/GP	FIA_UID.1	FIA_UID.1/CMGR
FIA_UAU.4/GP	No dependencies	
FIA_USB.1/GP	FIA_ATD.1	FIA_ATD.1/GP
FMT_MOF.1/GP	FMT_SMR.1 FMT_SMF.1	FMT_SMR.1/CMGR FMT_SMF.1/GP
FMT_MSA.2/GP	[FDP_ACC.1 or FDP_IFC.1] FMT_MSA.1 FMT_SMR.1	FDP_ACC.1/CMGR FMT_MSA.1/CMGR FMT_SMR.1/CMGR
FMT_MTD.1/GP	FMT_SMR.1 FMT_SMF.1	FMT_SMR.1/CMGR FMT_SMF.1/GP
FMT_SMF.1/GP	No dependencies	
FPT_TST.1/GP	No dependencies	
FCS_RNG.1/SCP	No dependencies	
FPT_FLS.1/SCP	No dependencies	
FPT_PHP.3/SCP	No dependencies	
FPT_RCV.3/SCP	AGD_OPE.1	AGD_OPE.1
FPT_RCV.4/SCP	No dependencies	
FRU_FLT.1/SCP	FPT_FLS.1	FPT_FLS.1/SCP

Table11. SFRs Dependencies

The dependency FIA_UID.1 of FMT_SMR.1/Installer is unsupported. This ST does not require the identification of the "installer" since it can be considered as part of the TSF.

The dependency FIA_UID.1 of FMT_SMR.1/ADEL is unsupported. This ST does not require the identification of the "deletion manager" since it can be considered as part of the TSF.

The dependency FAU_SAA.1 of FAU_ARP.1/CORE is unsupported. The dependency of FAU_ARP.1/CORE on FAU_SAA.1 assumes that a "potential security violation" generates an audit event. On the contrary, the events listed in FAU_ARP.1/CORE 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.

7.3.3.2 SARs dependencies

EALs in the CC consist of an appropriate combination of assurance components ad described in the CC Part 3. Each EAL includes no more than one component of each assurance family and all assurance dependencies of every component are addressed.
The composite ST augments tow assurance component ALC_DVS.2 and AVA_VAN.5, their dependencies are satisfied.

SARs	CC Dependencies	Satisfied Dependencies
ALC_DVS.2	No dependencies	
AVA_VAN.5	ADV_ARC.1 ADV_FSP.4 ADV_IMP.1 ADV_TDS.3 AGD_OPE.1 AGD_PRE.1 ATE_DPT.1	ADV_ARC.1 ADV_FSP.4 ADV_IMP.1 ADV_TDS.3 AGD_OPE.1 AGD_PRE.1 ATE_DPT.1

Table12. SARs Dependencies

7.3.4 Rationale for the security assurance requirements

EAL4 is required for this type of TOE and product since it is intended to defend against sophisticated attacks. This evaluation assurance level allows a developer to gain maximum assurance from positive security engineering based on good practices. EAL4 represents the highest practical level of assurance expected for a commercial grade product. In order to provide a meaningful level of assurance that the TOE and its embedding product provide an adequate level of defense against such attacks: the evaluators should have access to the low level design and source code. The lowest for which such access is required is EAL4.

7.3.5 ALC_DVS.2 sufficiency of security measures

Development security is concerned with physical, procedural, personnel and other technical measures that may be used in the development environment to protect the TOE and the embedding product. The standard ALC_DVS.1 requirement mandated by EAL4 is not enough. Due to the nature of the TOE and embedding product, it is necessary to justify the sufficiency of these procedures to protect their confidentiality and integrity. ALC_DVS.2 has no dependencies.

7.3.6 AVA_VAN.5 advanced methodical vulnerability analysis

The TOE is intended to operate in hostile environments. AVA_VAN.5 "Advanced methodical vulnerability analysis" is considered as the expected level for Java Card technology-based products hosting sensitive applications, in particular in payment and identity areas. AVA_VAN.5 has dependencies on ADV_ARC.1, ADV_FSP.1, ADV_TDS.3, ADV_IMP.1, AGD_PRE.1 and AGD_OPE.1. All of them are satisfied by EAL4.

8 TOE summary specification

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

8.1 Security Functionality

The following table provides a list of all security functions

No	Security function	Description
1	SF.AccessControl	enforces the access control
2	SF.Audit	Audit functionality
3	SF.Cryptography	Cryptographic key management & operation
4	SF.Authentication	Identification and authentication
5	SF.SecureManagement	Secure management of TOE resources
6	SF.Transaction	Transaction management
7	SF.Hardware	TSF of the underlying IC

8.1.1 SF.AccessControl

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

CARD CONTENT MANAGEMENT access control SFP on the following list of subjects, objects, operations and security attributes(see 7.1.7 FDP_ACC.1/CMGR and 7.1.7 FDP_ACF.1/CMGR) loading/installing a new application package on the card via a trusted channel (see 7.1.6 FTP_ITC.1/CM).

Subject/Object	Security attributes
S.OPEN	**None**
S.ISD	**AID, Privilege, Life Cycle State**
S.SD	**AID, Privilege, Life Cycle State**
O.CARD_CONTENT	**AID, Verified**
O.REGISTRY	**None**

Security attributes	Values
AID	**Application ID**
Privilege	**Application Privileges defined in [GP]**
Life Cycle State	**card: OP_READY, INITIALIZED, SECURED, CARD_LOCKED, TERMINATED** **security domain: INSTALLED, SELECTABLE, PERSONALIZED, LOCKED**
Verified	**Boolean (True or False)**

FIREWALL access control SFP on S.PACKAGE, S.JCRE, S.JCVM, O.JAVAOBJECT and all operations among subjects and objects covered by the SFP (see 7.1.1.1 FDP_ACC.2/FIREWALL and 7.1.1.1 FDP_ACF.1/FIREWALL)

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**

JCRMI access control SFP on S.CAD, S.JCRE, O.APPLET, O.REMOTE_OBJ, O.REMOTE_MTHD, O.ROR, O.RMI_SERVICE and all operations among subjects and objects covered by the SFP (see 7.1.4 FDP_ACC.2/JCRMI, 7.1.4 FDP_ACF.1/JCRMI)

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

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.(see 7.1.3 FDP_ACC.2/ADEL, 7.1.3 FDP_ACF.1/ADEL)

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

JCVM information flow control SFP on **S.JCVM, S.LOCAL, S.MEMBER, I.DATA and OP.PUT(S1, S2, I)**. (see 7.1.1.1 FDP_IFC.1/JCVM and 7.1.1.1 FDP_IFF.1/JCVM).

Subject/Information	Security attributes
S.JCVM	**Currently Active Context**

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

JCRMI information flow control SFP on **S.JCRE, S.CAD, I.RORD and OP.RET_RORD(S.JCRE,S.CAD,I.RORD)**. (see 7.1.4 FDP_IFC.1/JCRMI, 7.1.4 FDP_IFF.1/JCRMI)

Subject/Information	Security attributes
I.RORD	**ExportedInfo**

- **OP.RET_RORD(S.JCRE, S.CAD, I.RORD) is permitted only if the attribute**

ExportedInfo of I.RORD has the value "true" ([JCRE22], §8.5).
PACKAGE LOADING information flow control SFP on **S.INSTALLER, S.BCV, S.CAD and I.APDU** and all operations.(see 7.1.6 FDP_IFC.2/CM, 7.1.6 FDP_IFF.1/CM)

Subject/Information	Security attributes
S.INSTALLER	**SecureChannel, SecurityLevel**
S.BCV	**None**
S.CAD	**None**
I.APDU	**SecurityLevel, Verified**

Security attributes	Values
SecureChannel	**Boolean (True or False)**
SecurityLevel	**0 (none)** **1 (MAC)** **3 (Both Encryption and MAC)**
Verified	**Boolean (True or False)**

- An information flow between S.CAD (on the behalf of S.BCV) and S.INSTALLER is allowed if and only if **the following conditions are all satisfied:**
  - **SecureChannel of S.INSTALLER has the value 'True',**
  - **SecurityLevel of I.APDU meets SecurityLevel of S.INSTALLER**
  - **Verified of I.APDU is 'True'.**

Only the JCRE (S.JCRE) can modify the security attributes Selected Applet Context, modify the security attributes Registered Applets and Resident Packages and can modify the list of registered applets' AIDs (see 7.1.1.1 FMT_MSA.1/JCRE, 7.1.3 FMT_MSA.1/ADEL, 7.1.1.4 FMT_MTD.1/JCRE, 7.1.1.1 FMT_SMF.1/CORE, 7.1.1.1 FMT_SMR.1/CORE, 7.1.3 FMT_SMR.1/ADEL)

Only the JCRE (S.JCRE) can revoke the Returned References of O.RMI_SERVICE. (see 7.1.4 FMT_REV.1/JCRMI)
The TSF enforce the rules that determine the lifetime of remote object references. (see 7.1.4 FMT_REV.1/JCRMI)

Only the JCVM (S.JCVM) can modify the security attributes Currently Active Context and Active Applets and can modify the list of registered applets' AIDs (see 7.1.1.1 FMT_MSA.1/JCVM, 7.1.3 FMT_SMF.1/ADEL, 7.1.1.1 FMT_SMF.1/CORE, 7.1.1.1 FMT_SMR.1/CORE)

Its owner applet can modify the security attributes: ExportedInfo of O.REMOTE_OBJ and can modify the security attributes Returned References of O.RMI_SERVICE (see 7.1.4 FMT_MSA.1/EXPORT, 7.1.4 FMT_SMF.1/JCRMI, 7.1.4 FMT_MSA.1/REM_REFS, 7.1.4 FMT_SMR.1/JCRMI)

Card Administrator is allowed query and modify the SecureChannel, SecurityLevel and Verified and can modify, delete, and create the GlobalPlatform Registry. (see 7.1.6 FMT_MSA.1/CM, 7.1.7 FMT_MTD.1/GP, 7.1.6 FMT_SMF.1/CM, 7.1.6 FMT_SMR.1/CM)

OPEN can modify, delete, and create the security attributes listed in the following table. (see 7.1.7 FMT_MSA.1/CMGR, 7.1.7 FMT_SMR.1/CMGR, 7.1.7 FMT_SMF.1/CMGR, 7.1.7 FMT_SMF.1/GP)

Abilities	Security attributes
modify, delete, create	**AID, Privilege, and Life Cycle State**
modify	**Verified**

Only secure values are accepted for Verified and Authenticated. (see 7.1.7 FMT_MSA.2/GP)
Only secure values are accepted for the registered applets' AIDs. (see 7.1.1.4 FMT_MTD.3/JCRE)

The TSF 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(see 7.1.1.1 FMT_MSA.2/FIREWALL_JCVM)

Restrictive default values are used for the security attributes, which cannot be overwritten (see 7.1.7 FMT_MSA.3/CMGR, 7.1.1.1 FMT_MSA.3/FIREWALL, 7.1.1.1 FMT_MSA.3/JCVM, 7.1.3 FMT_MSA.3/ADEL, 7.1.4 FMT_MSA.3/JCRMI, 7.1.6 FMT_MSA.3/CM)

The TSF enforce the PACKAGE LOADING information flow control SFP when importing user data, controlled under the SFP, from outside of the TOE.
The TSF use the security attributes associated with the imported user data.
The TSF ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.
The TSF ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.
The TSF 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).(see 7.1.2 FDP_ITC.2/Installer, 7.1.2 FMT_SMR.1/Installer)

8.1.2 SF.Audit

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

TSF throw an exception, lock the card session, reinitialize the Java Card System and its data or mute the card when exceeding predefined number of secure channel establishment failure upon detection of a potential security violation.(7.1.1.3 FAU_ARP.1/CORE, 7.1.1.3 FPT_FLS.1/CORE)

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(), [JCAPI22] and ([JCRE22], §7.6.2)
- violation of the Firewall or JCVM SFPs,
- unavailability of resources,
- array overflow,
- other runtime errors related to applet's failure (like uncaught exceptions).

8.1.3 SF.Cryptography

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

Generation of DES keys with length of 64Bit. (see 7.1.1.2 FCS_CKM.1/CORE).
Generation of TDES keys with length of 128 and 192 Bit (see 7.1.1.2 FCS_CKM.1/CORE).
Generation of AES keys with length of 128, 192, and 256 Bit(see 7.1.1.2 FCS_CKM.1/CORE).
Generation of SEED keys with length of 128 Bit(see 7.1.1.2 FCS_CKM.1/CORE).
Generation of RSA keys with length from 512 to 2048 Bit(see 7.1.1.2 FCS_CKM.1/CORE).
Generation of EC over GF(p) keys with length from 112 to 521 Bit (see 7.1.1.2 FCS_CKM.1/CORE).

Label	Crypto Algorithm	Crypto Key Sizes	Standards
Protected RSA key generation	**RSA public and private keys computation algorithm, protected against side channel attacks**	**512 up to 2048 bits**	**FIPS PUB 140-2 ISO/IEC 9796-2 PKCS #1 V2.1**
	class KeyPair class KeyBuilder		**[JCAPI222]**
DES/3DES	**class KeyBuilder**	**DES: 56 effective bits (64bits) 3DES 2 keys: 112 effective bits (128bits) 3DES 3 keys: 168 effective bits (192bits)**	**[JCAPI222]**

AES	**class KeyBuilder**	**128, 192 and 256 bits**	**[JCAPI222]**
ECC	**class KeyBuilder**	**112 up to 521 bits**	**[JCAPI222]**
SEED	**class KeyBuilder**	**128 bits**	**[JCAPI222] [FICCS]**

Distribution of DES/TDES, AES, SEED keys with the method setKey of Java Card API (see 7.1.1.2 FCS_CKM.2/CORE).
Distribution of RSA keys with the method setExponent and setModulus of Java Card API(see 7.1.1.2 FCS_CKM.2/CORE).
Distribution of EC over GF(p) keys with the method setA, setB, setFieldFP, setG, setK, and setR of Java Card API (see 7.1.1.2 FCS_CKM.2/CORE).

Label	Crypto Key Distribution Method	Standards
RSA	**setExponent** **setModulus** **setDP1** **setDQ1** **setP** **setPQ** **setQ**	**[JCAPI222]**
DES/3DES	**setKey**	**[JCAPI222]**
AES	**setKey**	**[JCAPI222]**
ECC	**setA** **setB** **setFieldFP** **setG** **setK** **setR** **setS** **setW**	**[JCAPI222]**
SEED	**setKey**	**[JCAPI222]**
	setKey	**[FICCS]**

Management of DES/TDES, AES, SEED, ECC and RSA- keys with methods/commands defined in packages javacard.security and javacardx.crypto of Java Card API (see 7.1.1.2 FCS_CKM.3/CORE).

Label	Crypto Key Access Method	Standards
RSA keys	**methods packages javacard.security and javacardx,crypto**	**[JCAPI222]**
DES/3DES	**methods packages javacard.security and javacardx,crypto**	**[JCAPI222]**
AES	**methods packages javacard.security and**	**[JCAPI222]**

	javacardx,crypto	
ECC	**methods packages javacard.security and javacardx,crypto**	**[JCAPI222]**
SEED	**methods packages javacard.security and javacardx,crypto**	**[JCAPI222]**
	and koreanpackage	**[FICCS]**

Destruction of DES/TDES, AES, SEED, ECC and RSA- keys by physically overwriting the keys by method clearKey of Java Card API (see 7.1.1.2 FCS_CKM.4/CORE).

Label	Crypto Key Destruction Method	Standards
RSA keys	**clearKey() method**	**[JCAPI222]**
DES/3DES	**clearKey() method**	**[JCAPI222]**
AES	**clearKey() method**	**[JCAPI222]**
ECC	**clearKey() method**	**[JCAPI222]**
SEED	**clearKey() method**	**[JCAPI222]**
	clearKey() method	**[FICCS]**

Cryptographic algorithms and functionality(see 7.1.1.2 FCS_COP.1/CORE):

Label	Crypto Operations	Crypto Algorithm	Crypto Key Sizes	Standards
DES / 3DES operation	**encryption, decryption - in Cipher Block Chaining (CBC) mode - in Electronic Code Book (ECB) mode - in CBC-MAC operating modes**	**Data Encryption Standard (DES)**	**56 effective bits (64bits)**	**FIPS PUB 46-3 ISO/IEC 9797-1 ISO/IEC 10116**
		Triple Data Encryption Standard (3DES)	**2 keys: 112 effective bits (124bits) 3 keys: 168 effective bits (192bits)**	
RSA operation	**RSA recovery (encryption), RSA signature (decryption) without the Chinese Remainder Theorem,**	**Rivest, Shamir & Adleman's**	**512 up to 2048 bits**	**PKCS #1 V2.1**

	RSA signature (decryption) with the Chinese Remainder Theorem			
AES operation	**cipher operation, inverse cipher operation**	**Advanced Encryption Standard**	**128, 192 and 256 bits**	**FIPS PUB 197**
ECDSA operation	**general point addition, point expansion, point compression, public scalar multiplication, private scalar multiplication**	**Elliptic Curves Cryptography on GF(p)**	**112 up to 521 bits**	**IEEE 1363-2000, chapter 7 IEEE 1363a-2004**
SEED operations	**cipher operation, inverse cipher operation**		**128 bits**	**ISO/IEC 18033-3, IETF RFC 4269**
SHA-1 operation	**SHA-1 (secure hash function)**	**revised Secure Hash Algorithm (SHA-1)**	**N/A**	**FIPS PUB 180-1 FIPS PUB 180-2 SO/IEC 10118-3:1998**
SHA-256 operation	**SHA-256 (secure hash function)**	**revised Secure Hash Algorithm (SHA-256)**	**N/A**	**FIPS PUB 180-1 FIPS PUB 180-2 SO/IEC 10118-3:1998**

8.1.4 SF.Authentication

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

Get Data, Select Applet and Manage Channel are possible before authentication (see 7.1.6 FIA_UID.1/CM, 7.1.7 FIA_UID.1/CMGR, 7.1.7 FIA_UAU.1/GP).

The TSF terminate the card when 10 unsuccessful Secure Channel Establishment (see 7.1.7 FIA_AFL.1/GP).
The TSF block the PIN when 3 unsuccessful CVM (see 7.1.7 FIA_AFL.1/GP).

The TSF prevent reuse of authentication data related to the Secure Channel Establishment.(see 7.1.7 FIA_UAU.4/GP)

8.1.5 SF.SecureManagement

The TSF provide a secure management of TOE resources:

The TSF makes any previous information content of a resource unavailable upon (see 7.1.1.1 FDP_RIP.1/OBJECTS, 7.1.1.2 FDP_RIP.1/APDU, 7.1.1.2 FDP_RIP.1/bArray, 7.1.1.2 FDP_RIP.1/TRANSIENT, 7.1.1.2 FDP_RIP.1/ABORT and 7.1.1.2 FDP_RIP.1/KEYS, 7.1.3 FDP_RIP.1/ADEL, 7.1.5 FDP_RIP.1/ODEL):
- allocation of class instances, arrays, and the APDU buffer,
- de-allocation of bArray object, any transient object, any reference to an object instance created during an aborted transaction, cryptographic buffer (D.CRYPTO), applet instances and/or packages when one of the deletion operations, and the objects owned by the context of an applet instance which triggered the execution of the method

The TSF monitors user data checksum of D.APP_CODE, D.APP_I_DATA, D.PIN, D.APP_KEYs for integrity errors. If an error occurs, the TSF take actions defined in FAU_ARP.1/CORE (see 7.1.1.3 FDP_SDI.2/CORE).

The TSF ensures that any user and subject are unable to observe operations on PIN code and cryptographic keys by TSF (see 7.1.1.3 FPR_UNO.1/CORE).

The TSF preserve a secure state when the following types of failures occur:

- the installer fails to load/install a package/applet as described in [JCRE22] §11.1.4.(see 7.1.2 FPT_FLS.1/Installer)
- the applet deletion manager fails to delete a package/applet as described in [JCRE22], §11.3.4.(see 7.1.3 FPT_FLS.1/ADEL)
- the object deletion functions fail to delete all the unreferenced objects owned by the applet that requested the execution of the method.(see 7.1.5 FPT_FLS.1/ODEL)

CAP files, the bytecode and its data arguments are consistently interpreted using the following rules (see 7.1.1.3 FPT_TDC.1/CORE):

- the rules defined in [JCVM22] specification,
- the API tokens defined in the export files of reference implementation,
- the rules defined in [VGP]
- the rules defined in the [ISO7816], [ISO14443], [EMV42] and [EMVCL201]

TSF 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).
- Card Administrator: AID, Life Cycle State, Privilege, ISD Keys
- Application Provider: AID, Life Cycle State, Privilege, SD Keys
- End-user (Card Holder): AID, Privilege, CVM State, Retry Limit, Retry Counter.

(see 7.1.1.4 FIA_ATD.1/AID, 7.1.1.4 FIA_UID.2/AID, 7.1.1.4 FIA_USB.1/AID, 7.1.7 FIA_ATD.1/GP and 7.1.7 FIA_USB.1/GP).

The TSF run a suite of self-tests during initial start-up to demonstrate the correct operation of the TSF, to verify the integrity of parts of TSF data and to verify the integrity of stored parts of TSF. (see 7.1.7 FPT_TST.1/GP).

The TSF enforce the generation of evidence of origin for transmitted application packages at all times. The TSF 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. The TSF provide a capability to verify the evidence of origin of information to recipient given immediate verification of origin.(see 7.1.6 FCO_NRO.2/CM)

The TSF enforce the PACKAGE LOADING information flow control SFP to receive user data in a manner protected from modification, deletion, insertion, and replay errors(see 7.1.6 FDP_UIT.1/CM)

The TSF restrict the ability to disable, enable, modify the behavior of the functions listed in the following table to Card Administrator. (see 7.1.7 FMT_MOF.1/GP)

8.1.6 SF.Transaction

The TSF permits the rollback of operations OP.JAVA, OP.CREATE on object O.JAVAOBJECT. These operations can be rolled back within the calls: select(), deselect(), process(), install() or uninstall() call, notwithstanding the restrictions given in [JCRE22], §7.7, within the bounds of the Commit Capacity ([JCRE22], §7.8), and those described in [JCAPI22].(see 7.1.1.2 FDP_ROL.1/FIREWALL, 7.1.2 FPT_RCV.3/Installer)
Transactions are a service offered by the APIs to applets. It is also used by some APIs to guarantee the atomicity of some operation. This mechanism is either implemented in Java Card platform or relies on the transaction mechanism offered by the underlying platform.
Some operations of the API are not conditionally updated, as documented in [JCAPI22] (see for instance, PIN-blocking, PIN-checking, update of Transient objects).

8.1.7 SF.Hardware

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

Protection against Physical Manipulation (see 7.1.8 FCS_RNG.1/SCP, 7.1.8 FPT_PHP.3/SCP)

The TSF ensure the operation of JCS and CM capabilities when the following failures occur: lack of EEPROM, random number generator failure, crypto co-processor failure, RAM read/write failure, card tearing, and power failure.(see 7.1.8 FRU_FLT.1/SCP)

When automated recovery from none is not possible, the TSF enter a maintenance

mode where the ability to return the TOE to a secure state is provided. For all cases, the TSF ensure the return of the TOE to a secure state using automated procedures. The functions provided by the TSF to recover from failure or service discontinuity ensure that the secure initial state is restored without exceeding 0% for loss of TSF data or objects within the TSC. The TSF provide the capability to determine the objects that were or were not capable of being recovered. (see 7.1.8 FPT_RCV.3/SCP)

The TSF ensure that reading from and writing to static and objects' fields interrupted by power loss have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state. (see 7.1.8 FPT_RCV.4/SCP)

The TSF preserve a secure state when the following types of failures occur:

- exposure to operating conditions which may not be tolerated according to the requirement FRU_FLT.1 and where therefore a malfunction could occur.(see 7.1.8 FPT_FLS.1/SCP)

9 Annexes

9.1 References

[CCPART1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, July 2009, Version 3.1 Revision 3 Final, CCMB-2009-07-001

[CCPART2] Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements, July 2009, Version 3.1 Revision 3 Final, CCMB-2009-07-002

[CCPART3] Common Criteria for Information Technology Security Evaluation, Part 2: Security assurance requirements, July 2009, Version 3.1 Revision 3 Final, CCMB-2009-07-003

[COMP-EVAL] Composite product evaluation for Smart Cards and similar devices, September 2007, Version 1.0 Revision 1, CCDB-2007-09-001

[JCSPP] Java Card™ System Protection Profile Open Configuration, Version 2.6, 19 April 2010

[PP0035] Security IC Platform Protection Profile, Version 1.0, 15 July 2007

[PP JCS] Java Card Protection Profile Collection, Version 1.0b, August 2003, registered and certified by the French certification body (ANSSI) under the following references: [PP/0303] “Minimal Configuration”, [PP/0304] ″Standard 2.1.1 Configuration”, [PP/0305] “Standard 2.2 Configuration” and [PP/0306] “Defensive Configuration”.

[ICST] SA23YR48B / SB23YR48B / SA23YR80B / SB23YR80B SECURITY TARGET - PUBLIC VERSION, Rev 02.01, issued November 2009

[JCRE22] Java Card Platform, version 2.2 Runtime Environment (Java Card RE) Specification. June 2002. Published by Sun Microsystems, Inc.

[JCVM22] Java Card Platform, version 2.2 Virtual Machine (Java Card VM) Specification. June 2002. Published by Sun Microsystems, Inc.

[JCRE222] Java Card Platform, version 2.2.2 Runtime Environment (Java Card RE) Specification. March 2006. Published by Sun Microsystems, Inc.

[JCVM222] Java Card Platform, version 2.2.2 Virtual Machine (Java Card VM) Specification. Beta release, October 2005. Published by Sun Microsystems, Inc.

[JCAPI222] Java Card Platform, version 2.2.2 Application Programming Interface, March 2006. Published by Sun Microsystems, Inc.

[GP] GlobalPlatform Card Specification, Version 2.1.1, March 2003

[VGP] Visa GlobalPlatform 2.1.1 Card Implementation Requirements, Version 2.0, July 2007

[JAVASPEC] The Java Language Specification. Third Edition, May 2005. Gosling, Joy, Steele and Bracha. ISBN 0-321-24678-0.

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

[JCBV] Java Card Platform, version 2.2 Off-Card Verifier. June 2002. White paper. Published by Sun Microsystems, Inc.

[FICCS] Finance IC card standard revision - Open platform – October 2010

[EMV42] EMV Integrated Circuit Card Specifications for Payment Systems Book 1 Application Independent ICC to Terminal Interface Requirements Version 4.2 June 2008
[ISO7816] ISO/IEC 7816-3 Part 3: Cards with contacts – Electrical interface and transmission protocols 2006-11-01
[EMVCL201] EMV
Contactless Specifications for
Payment Systems EMV Contactless Communication
Protocol Specification Version 2.0.1 July 2009
[ISO14443] ISO/IEC 14443-3 Part 3: Initialization and anticollision 1999-06-11

9.2 Terms and definitions

Application Protocol Data Unit(APDU)	Standard communication command protocol between smartcard and CAD
Application (Applet)	The name is given to a Java Card technology-based user application. An application is the basic piece of code that can be selected for execution from outside the card. Each application on the card is uniquely identified by its AID.
Application instance	Instance of an Executable Module after it has been installed and made selectable.
Application Protocol Data Unit(APDU)	Standard communication messaging protocol between a card accepting device and a smart card
Application Provider	Entity that owns an application and is responsible for the application's behavior
bytecode verifier	The bytecode verifier is the software component performing a static analysis of the code to be loaded on the card. It checks several kinds of properties, like the correct format of CAP files and the enforcement of the typing rules associated to bytecodes. If the component is placed outside the card, in a secure environment, then it is called an off-card verifier. If the component is part of the embedded software of the card it is called an on-card verifier.
CAD	Card Acceptance Device, or card reader. The device where the card is inserted, and which is used to communicate with the card.
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 Content	Code and Application information (but not Application data) contained in the card that is under the responsibility of the OPEN e.g. Executable Load Files, Application instances, etc
Cardholder	The end user of a card
Cardholder Verification Method (CVM)	A method to ensure that the person presenting the card is the person to whom the card was issued
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
Card Manager	Generic term for the 3 card management entities of a GlobalPlatform card i.e. the OPEN, Issuer Security Domain and the Cardholder Verification Method Services provider
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
DAP Block	Part of the Load File used for ensuring Load File Data Block verification
DAP Verification	A mechanism used by a Security Domain to verify that a Load File Data Block is authentic
Default applet	The applet that is selected after a card reset
Domain	Issuer Security Domain : On-card entity providing support for the control, security, and communication requirements of the Card Issuer Security Domain : On-card entity providing support for the control, security, and communication requirements of the Application Provider

ES(Embedded Software)	it is defined as the software embedded in the Smart Card Integrated Circuit. The ES may be in any part of the non-volatile memories of the Smart Card IC.
Executable File	Actual on-card container of one or more Executable Modules. It may reside in immutable persistent memory or may be created in mutable persistent memory as the resulting image of an Executable Load File.
Executable Load File	An Executable File that is in transit to the smart card.
Executable Module	Contains the on-card executable code of a single application present within an Executable Load File
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.
GP	Global Platform, GP is an organization that has been established by leading companies from the payments and communications industries, the government sector and the vendor community, and is the first to promote a global infrastructure for smart card implementation across multiple industries. Its goal is to reduce barriers hindering the growth of cross-industry, multiple Application smart cards. The smart card issuers will continue to have the freedom to choose from a variety of cards, terminals and back-end systems.
GlobalPlatform Registry	A container of information related to Card Content management
IC	Integrated Circuit , Electronic component(s) designed to perform processing and/or memory functions
Installer	The installer is the on-card application responsible for the installation of applets on the card. It may perform (or delegate) mandatory security checks according to the card issuer policy (for bytecode-verification, for instance), loads and link package s (CAP file (s)) on the card to a suitable form for the JCVM to execute the code they contain. It is a subsystem of what is usually called “card manager”; as such, it can be seen as the portion of the card manager that belongs to the TOE. The installer has an AID that uniquely identifies him, and may be implemented as a Java Card applet. However, it is

	granted specific privileges on an implementation-specific manner
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.
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.
JCRMI	Java Card Remote Method Invocation is the Java Card System, version 2.2.2, mechanism enabling a client application running on the CAD platform to invoke a method on a remote object on the card. Notice that in Java Card System, version 2.1.1, the only method that may be invoked from the CAD is the **process** method of the **applet** class
JCVM	The embedded interpreter of bytecodes. The JCVM is the component that enforces separation between applications (firewall) and enables secure data sharing.
Issuer Security Domain	On-card entity providing support for the control, security, and communication requirements of the Card Issuer
logical channel	A logical link to an application on the card. A new feature of the Java Card System, version 2.2.2, that enables the opening of up to four simultaneous sessions with the card, one per logical channel. Commands issued to a specific logical channel are forwarded to the active applet on that logical channel.
Object deletion	The Java Card System, version 2.2.2, mechanism

	ensures that any unreferenced persistent (transient) object owned by the current context is deleted. The associated memory space is recovered for reuse prior to the next card reset.
Open Platform Environment (OPEN)	The central on-card administrator that owns the GlobalPlatform Registry
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.
Transient object	An object whose contents are not preserved across CAD sessions. The contents of these objects are cleared at the end of the current CAD session or when a card reset is performed. Writes to the fields of a transient object are not affected by transactions.

9.3 Abbreviated terms

AID	Application Identifier
APDU	Application Data Protocol Unit
API	Application Programming Interface
ATR	Answer-to-Reset
CAD	Card Acceptance Device
CAP	Converted Applet
CC	Common Criteria
CVM	Cardholder Verification Method
DAP	Data Authentication Pattern
DES	Data Encryption Standard
EAL	Evaluation Assurance Level
EMV	Europay, MasterCard, and Visa; used to refer to the ICC Specifications for Payment Systems
GP	Global Platform
ICC	Integrated Circuit Card
ISD	Issuer Security Domain
ISO	International Organization for Standardization
JCRE	Java Card Runtime Environment
JCVM	Java Card Virtual Machine
JCAPI	Java Card API
JCS	Java Card System
MAC	Message Authentication Code
OPEN	Open Platform Environment

OS	Operating System
OSP	Organisational Security Policy
PIN	Personal Identification Number
PP	Protection Profile
RAM	Random Access Memory
ROM	Read-only Memory
RSA	Rivest / Shamir / Adleman asymmetric algorithm
SCP	Secure Channel Protocol
SD	Security Domain
ST	Security Target
SIO	An object of a class implementing a shareable interface
TOE	Target Of Evaluation
TSF	TOE Security Functionality
VM	Virtual Machine