THALES

gemalto
a Thales company

Security Target

Public version

MultiApp Essential v1.1

Common Criteria / ISO 15408
EAL 5+

Rev 2.3p
September 15th, 2019.

CONTENT

FIGURES

TABLES

1 SECURITY TARGET INTRODUCTION

1.1 SECURITY TARGET REFERENCE

Title :	MultiApp Essential v1.1: JCS Security Target
Version :	2.3p
ST Reference :	D1462421
IT Security Evaluation scheme :	Applus
IT Security Certification scheme :	Centro Criptológico Nacional Organismo de Certificación (OC-CCN)

1.2 TOE REFERENCE

Product Technical Name :	MultiApp Essential v1.1
Security Controller :	CENTAURUS
TOE Name :	MultiApp Essential v1.1 Platform
TOE Version :	1.1
TOE Documentation	See section Guidance [AGD]
Composition elements:	
IC Product name:	CENTAURUS
IC Reference:	CENTAURUS_FB_04
IC Hardware revision:	F
IC Platform ROM Firmware Revision	B
IC Manufacturer	INVIA

The TOE identification is provided by the Card Production Life Cycle Data (CPLCD) of the TOE. These data are available by executing a dedicated command (see section 2.4.1)

The tag identity provides information on the product and allow to identify each product configuration.

In this case, there are two possible configurations on this product: Full and Light. The only feature making the difference between each other is as follows:

- MultiApp Essential v1.1 Full Configuration
  - OBKG available
- MultiApp Essential v1.1 Light Configuration
  - OBKG not available

The identification of these two configurations are specified on section 2.4.1.

1.3 SECURITY TARGET OVERVIEW

The Target of Evaluation (TOE) addressed by the current Security Target is a product comprising hardware and software corresponding to a Java Card platform operating system in Open Configuration.

- The TOE is conformed of:
  - The underlying Integrated Circuit
  - The MultiApp Essential v1.1 platform (JavaCard platform)
  - The associated guidance documentation

The MultiApp Essential v1.1 product also includes 3 applets.

Evaluation is performed using a composite approach defined in [JIL_CPE].
The IC is evaluated in conformance with [PP-IC-0084] according to [ST-IC] and certified in [CR-IC].

The main objectives of this ST are:

- To introduce TOE and the JCS Platform,
- To define the scope of the TOE and its security features,
- To describe the security environment of the TOE, including the assets to be protected and the threats to be countered by the TOE and its environment during the product development, production and usage.
- To describe the security objectives of the TOE and its environment supporting in terms of integrity and confidentiality of application data and programs and of protection of the TOE.
- To specify the security requirements which includes the TOE security functional requirements, the TOE assurance requirements and TOE security functions.

1.4 REFERENCES

1.4.1 External References

[CC]	Common Criteria references
[CC-1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017.
[CC-2]	Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, CCMB-2017-04-002, Version 3.1 Revision 5, April 2017.
[CC-3]	Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.
[CEM]	Common Methodology for Information Technology Security Evaluation Evaluation Methodology CCMB-2017-04-004, version 3.1 rev 5, April 2017
[JIL_CPE]	Joint Interpretation Library: Composite product evaluation for Smart Cards and similar devices, Version 1.5.1 May 2018
[ACM]	SOG-IS Crypto Evaluation Scheme - Agreed Cryptographic Mechanisms, version 1.1 June 2018
[PP]	**Protection profiles**
[PP-IC-0084]	Security IC Platform Protection Profile with augmentation Packages– BSI-CC-PP-0084-2014, version 1.0.
[PP-JCS-Open]	Java Card System – Open Configuration Protection Profile ANSSI-PP-2010-03M01, Version 3.0, May 2012
[RGS-B1]	Référentiel Général de sécurité version 2 Annexe B1 Mécanismes cryptographiques, règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques; version 2.0.3 du 21 février 2014
[ST-JCS]	This document : Security Reference (D1462421)
[IC]	**IC References**
[ST-IC]	[ST-IC-CENT]
[ST-IC-CENT]	Security Target Lite for CENTAURUS (Microcontroller CENTAURUS_FB_04) Reference: CENTAURUS_F_ST_Lite Rev. 1.0, April 16, 2018
[CR-IC]	[CR-IC-CENT]
[CR-IC-CENT]	Rapport de certification ANSSI-CC-2018/51 Microcontroller CENTAURUS_FB_04 (Révision du matériel F) 29 novembre 2018.
[NIST]	**NIST references**
[FIPS180-4]	*FIPS PUB 180-4, Federal Information Processing Standards Publication* *SECURE HASH STANDARD (SHS)* Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899-8900, August 2015
[FIPS197]	*Federal Information Processing Standards Publication 197 ADVANCED ENCRYPTION STANDARD (AES),* 2001 November 26
[SP800-67]	NIST Special Publication 800-67 Revision 2 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, November 2017
[ISO]	**ISO references**

[ISO9796-2]	*ISO/IEC 9796-2:2010: Information technology – Security techniques – Digital Signature Schemes giving message recovery – Part 2: Integer factorization based mechanisms,* Third edition 2010-12-15
[ISO9797-1]	*ISO/IEC 9797-1:2011: Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher,* Second edition 2011-03-01
[GP]	**Global Platform references**
[GP221]	GlobalPlatform Card Specification Version 2.2.1 Public Release January 2011 Document Reference: GPC_SPE_034
[GP221Amend D]	GlobalPlatform Card Technology Secure Channel Protocol '03' Specification 2.2 Amendment D Version 1.1.1, July 2014 Document Reference: GPC_SPE_014
[GP221Amend E]	GlobalPlatform Card Technology Security Upgrade for Card Content Management Card Specification v2.2 -- Amendment E Version v1.0.1, July 2014 Document Rference: GPC_SPE_042
[GP221 MpGd]	GlobalPlatform Card Mapping Guidelines of Existing GP v2.1.1 Implementation on v2.2.1 Version 1.0.1, January 2011 Document Reference: GPC_GUI_003
[GP221 Id Config]	GlobalPlatform – ID Configuration Version 1.0, December 2011

[JCS]	Javacard references
[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. Second Edition. June 2000. 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.
[JCRE304]	Java Card 3 Platform Runtime Environment Specification, Classic Edition, Version 3.0.4 – September 2011 – Published by Oracle
[JCVM304]	Java Card 3 Virtual Machine Specification, Classic Edition, Version 3.0.4-– September 2011 – Published by Oracle
[JCAPI304]	Java Card 3 Application Programming Interface Specification, Classic Edition, Version 3.0.4-– September 2011 – Published by Oracle
[PP JCS]	Java Card System 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”.

1.4.2 Internal References [IR]

[AGD]	MulitiApp Essential V1.1 Software – Guidance documentation
[AGD-PRE]	MultiApp Essential v1.1: AGD_PRE document - Javacard Platform Ref. D1462570, Rev 1.8, April 9 2019
[AGD-OPE]	MultiApp Essential v1.1: AGD_OPE document - Javacard Platform, Ref. D1462568, Rev 1.4, March 8 2019
[AGD-Ref]	MultiApp Essential Operating System Reference Manual, Ref. D1352558E, March 8 2019
[Applet guidance-1]	Guidance for secure application development on Multiapp platforms, D1390326 Rev. A01, March 2018
[Applet guidance-2]	Verification process of Gemalto non sensitive applet, D1390670 Rev. A02, October 2018
[Applet guidance-3]	Verification process of Third Party non sensitive applet, D1390671 Rev. A02, October 2018
[Applet guidance-4]	Rules for applications on Multiapp certified product, D1390963 Rev. 1.3, October 2018

1.5 ACRONYMS AND GLOSSARY

AES	Advanced Encryption Standard
APDU	Application Protocol Data Unit
API	Application Programming Interface
CAD	Card Acceptance Device
CC	Common Criteria
CPU	Central Processing Unit
DES	Data Encryption Standard
EAL	Evaluation Assurance Level
ECC	Elliptic Curve Cryptography
EEPROM	Electrically-Erasable Programmable Read-Only Memory
ES	Embedded Software
GP	Global Platform
IC	Integrated Circuit
IT	Information Technology
JCRE	JavaCard Runtime Environment
JCS	JavaCard System
JCVM	JavaCard Virtual Machine
NVM	Non-Volatile Memory
OP	Open Platform
PIN	Personal Identification Number
PP	Protection Profile
RMI	Remote Method Invocation
RNG	Random Number Generator
ROM	Read-Only Memory
RSA	Rivest Shamir Adleman
SAR	Security Assurance Requirement
SC	Smart Card
SCP	Secure Channel Protocol
SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
ST	Security Target
TOE	Target Of Evaluation
TSF	TOE Security Functionality

2 TOE OVERVIEW

2.1 INTRODUCTION

The TOE is an open Java Card and GlobalPlatform operating system (IC and OS) intended to host a set of Java Card applications. It provides cryptographic services for symmetric (DES, 3DES, AES) operations, asymmetric (RSA) operations and hash (SHA) operations, Moreover, it also provides random number generation and integrity check features.
All these security features are in the scope of the evaluation except the algorithms and key sizes not agreed by the document Agreed Cryptographic Mechanisms [ACM]. Not evaluated security features are:

- DES
- RSA with key sizes < 1900 bits
- SHA-1

The TOE has two possible configurations:

- MultiApp Essential v1.1 Full Configuration. The OBKG cryptographic feature is available.
- MultiApp Essential v1.1 Light Configuration. The OBKG cryptographic feature is not available.

The OBKG provides on board key generations (RSA).

Java Card RMI is not implemented in the TOE.

2.2 TOE TYPE

The TOE type of this evaluation is a Smart Card Platform (IC and OS) with the Java Card System.

The Java Card technology combines a subset of the Java programming language with a runtime environment optimized for smart cards and similar small-memory embedded devices [JCVM304]. The Java Card platform is a smart card platform enabled with Java Card technology (also called a “Java card”). This technology allows for multiple applications to run on a single card and provides facilities for secure interoperability of applications. Applications for the Java Card platform (“Java Card applications”) are called applets.

This TOE provides the security of an EAL5+ evaluated card with the flexibility of an open platform.
It allows for the loading of applets before or after the issuance of the card. These applets MAY or MAY NOT be evaluated on this platform.

The applications using only certified applets will BE certified even if NOT-certified applets are loaded on the platform.

The applications using a NOT-certified applet will NOT BE certified.

The Issuer can forbid the loading of applets before or after the issuance of the card.

2.3 PRODUCT ARCHITECTURE

The TOE is part of the MultiApp Essential v1.1 smartcard Product. This smartcard contains the software dedicated to the operation of:

- The MultiApp Essential V1.1 Platform, which supports the execution of the personalized applets and provides the smartcard administration services. It is conformant to JavaCard 3.0.4 and GP 2.2.1 standards.
- The identity applets: IAS Classic v3.0, Microsoft Plugin Play, MOC Client v1.1 & Server v1.1 (These applications could be removed based on customer needs and they are out of scope of the evaluation).

- Additionally, other applets – not determined at the moment of the present evaluation – may be loaded on the smartcard before or after issuance.
- A cryptographic library developed by Gemalto.

Therefore, the architecture of the smartcard software and application data can be represented as follows:

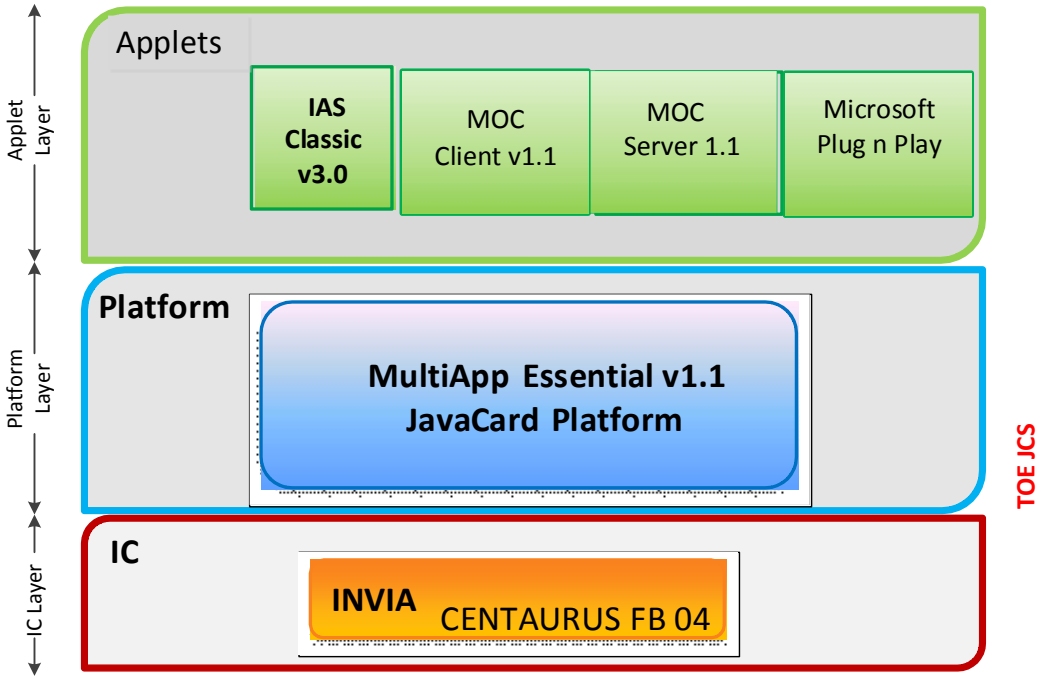

Figure 1: MultiApp Essential V1.1 smartcard architecture

The IAS Classic V3 applet, MOC Server v1.1 and MOC Client v1.1 applets and the MultiApp Essential V1.1 Java Card platform, are located in flash code area.

All the data (related to the applets or to the Java Card platform) are located in flash data area. The separation between these data is ensured by the Java Card firewall as specified in [JCRE304].

2.4 TOE DESCRIPTION

The MultiApp Essential V1.1 platform is a smart card operating system (IC and OS) that complies with two major industry standards:

- Sun's Java Card 3.0.4, which consists of the Java Card 3.0.4 Virtual Machine [JCVM304], the Java Card 3.0.4 Runtime Environment [JCRE304] and the Java Card 3.0.4 Application Programming Interface [JCAPI304].
- The Global Platform Card Specification version 2.2.1 [GP221].

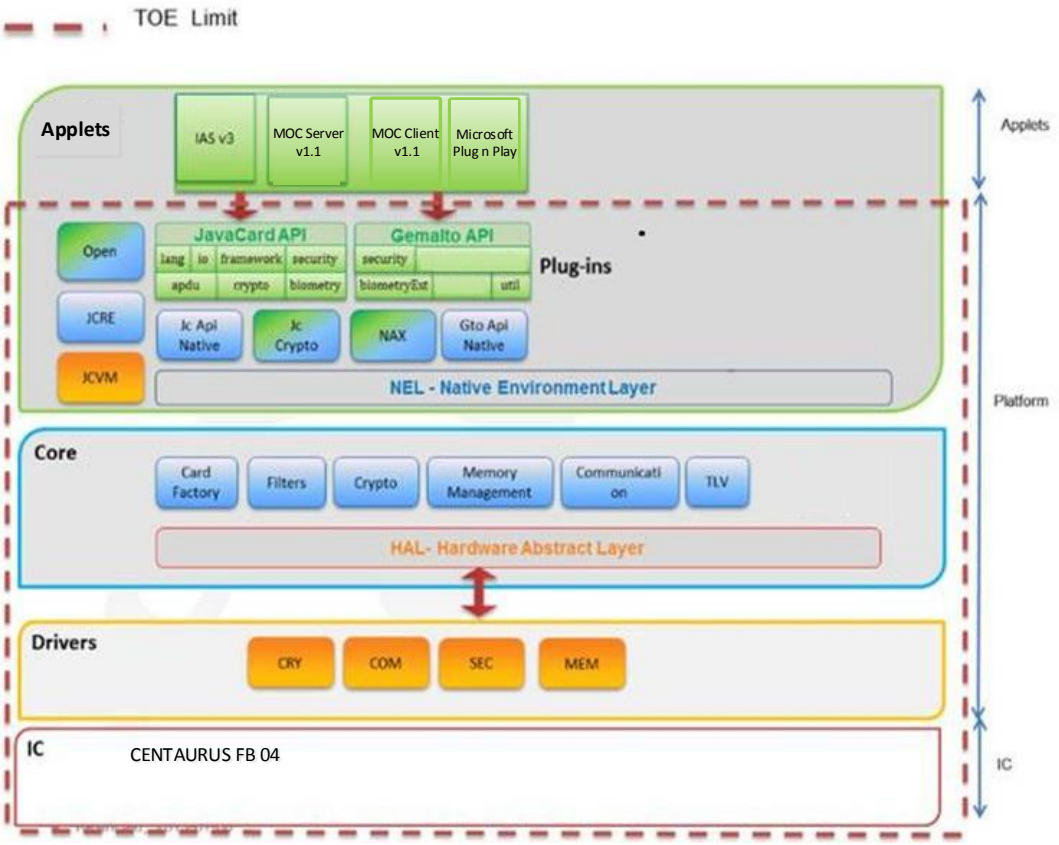

Figure 2: MultiApp Essential V1.1 Java Card platform architecture

As described in Figure 2, the MultiApp Essential V1.1 platform contains the following components:

- **The Core Layer**

  It provides the basic card functionalities (memory management, I/O management and cryptographic primitives) with native interface with the underlying IC. The cryptographic features implemented in the native layer encompass the following algorithms:

  - DES, 3DES (ECB, CBC)
  - RSA up to 3072 (CRT method), 2048 (Std method)
  - AES 128, 192, 256
  - SHA-1, SHA-224, SHA-256, SHA-384, SHA-512
  - DH up to 2048
  - Optional feature, OBKG 3K (CRT)

- PRNG
- CRC16
- CRC32 (mandatory for GP 2.2.1 ID config)

(*) DES, SHA-1 and RSA keys <1900 bits are algorithms supported by the product but they are not conformant with SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms [ACM] and are out of the scope of the CC evaluation.

- **The Plug-ins layer**

  - The **Javacard Runtime Environment**

    It conforms to [JCRE304] and provides a secure framework for the execution of the Java Card programs and data access management (firewall).

    Among other features, multiple logical channels are supported, as well as extradition, DAP, Delegated management and SCP03.

  - The **Javacard Virtual Machine**

    It conforms to [JVCM304] and provides the secure interpretation of bytecodes.

  - The **API**

    It includes the standard Java Card API [JCAPI304] and the Gemalto proprietary API.

  - The **Global Platform Issuer Security Domain**

    It conforms to [GP221] and provides card, key and applet management functions (contents and life-cycle) and security control.

The MultiApp Essential V1.1 platform provides the following services:

- Initialization of the Card Manager and management of the card life cycle
- Secure loading and installation of the applets under Security Domain control
- Deletion of applications under Issuer Security Domain control
- Extradition services to allow several applications to share a dedicated security domain
- Secure operation of the applications through the API
- Management and control of the communication between the card and the CAD
- Card basic security services as follows:
    - Checking environmental operating conditions using information provided by the IC
    - Checking life cycle consistency
    - Ensuring the security of the PIN and cryptographic key objects
    - Generating random numbers
    - Handling secure data object and backup mechanisms
    - Managing memory content
    - Ensuring Java Card firewall mechanism

2.4.1 Product and TOE Identification

As mentioned before, the TOE identification is provided by the Card Production Life Cycle Data (CPLCD) of the TOE. These data are available by executing a dedicated command described in [AGD-OPE] and here below:

The card production life cycle (CPLC) data is described in the GlobalPlatform 2.2.1 specification.

The TOE (MultiApp Essential v1.1 Platform) can be identified through the Get Data Command response with tag "0103", as follows:

Name	Description	Value
Gemalto Family Name	Java Card	B0
Gemalto OS name	MultiApp Essential v1.1	8D
Gemalto Mask Number	ISS008 (S8 Centaurus) – Gov BU ROM	5A
Gemalto Product Name	MultiApp Essential v1.1 – Config Full	57
	MultiApp Essential v1.1 – Config Light	5C
Gemalto Flow version (certified product configuration)	MultiApp Essential v1.1 – Config Full	11 (value HEX)
	MultiApp Essential v1.1 – Config Light	31 (value HEX)
Gemalto Filter version	Filter release version	00
Chip Manufacturer Code	INVIA	12 90
Chip Version	IC_TYPE = '0005': Value provided by ISSM	00 05
Chip BPU version		00 05
PDM Product ID		00 00 00
PDM Customer Item ID		00 00 00
Crypto Configuration	20 5D: Without Thai ID Configuration (Default) 24 5D: Thai ID Configuration	20 5D
Feature Configuration	Modularity feature not embedded	00
Biometry Configuration + RFU		00
Tag ID	1st byte: Baseline version 2nd byte: Dev path version	01 32

Table 1: TOE Identification Get Data with tag 0103

Using Get data command with tag "9F7F":

Date element	Length	Value
IC Fabricator	2	1290
IC Type	2	0005
OS Identifier	2	1981
OS Release Date	2	8165
OS Release Level	2	0101

Table 2: TOE Identification Get Data with tag 9F7F

The TOE is represented in the following table:

Integrated Circuit	
Product name	CENTAURUS
Reference	CENTAURUS_FB_04
Hardware revision	F
Platform ROM Firmware Revision	B
Platform FLASH Firmware Revision	04
BIOS	Version 1.0-739
LOADER	Version 2.1
Java Card Operating System	
OS name	MultiApp Essential v1.1
Mask reference label	LBL25 Checkpoint 1.32

Table 3: TOE composite

2.4.2 TOE BOUNDARIES

The Target of Evaluation (TOE) is the JCS open platform of the MultiApp Esssential V1.1 product. It is defined by:

- The Java Platform 3.0.4 based on JLEP3 Operating System
- The underlying Integrated Circuit

The applications, IAS classic V3, MOC Server v1.1 and MOC Client v1.1, stored in Flash mask in code area in MultiApp Essential V1.1, are outside the TOE.

The Applets loaded pre issuance or post issuance are outside the TOE, Other smart card product elements, (such as holograms, magnetic stripes, security printing) are outside the scope of this Security Target.

Java Card RMI is not implemented in the TOE.

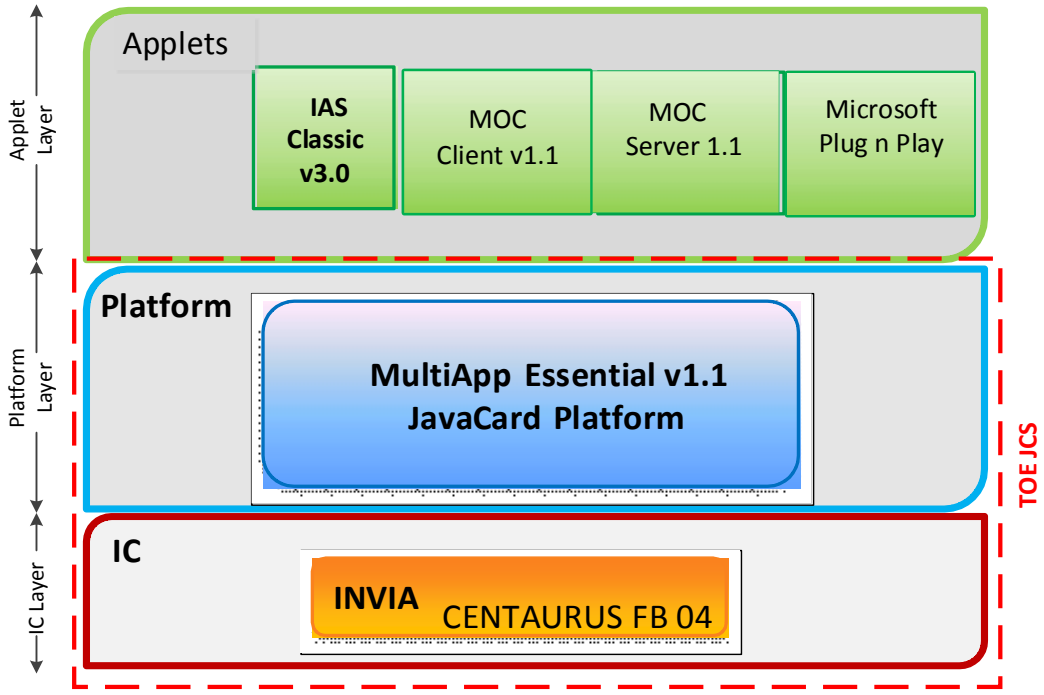

Figure 3: JCS TOE boundaries

2.4.3 TOE DELIVERY

As a summary description of how the parts of the TOE are delivered to the final customer, the MultiApp Essential v1.1 application is delivered mainly in form of a smart card or module. The form factor is packaged on Gemalto's manufacturing facility and sent to final customer premises.

The product is sent to final customer by standard transportation respecting Gemalto Transport Security Policies.

The different guides accompanying the TOE and parts of the TOE are the ones specified in [AGD] section. They are delivered in form of electronic documents (*.pdf) by Gemalto's Technical representative via a secure file sharing platform download action.

Item type	Item	Reference/Version	Form of delivery
Software and Hardware	MultiApp Essential v1.1	MULTIAPPESSV11_CODE_LBL25, checkpoint 1.32	Smart card or module
Document	MultiApp Essential Operating System Reference Manual	D1352558E March 8, 2019	Electronic document via secure file download
Document	Guidance for secure application development on Multiapp platforms	D1390326 A01 March 2018	Electronic document via secure file download
Document	Verification process of Gemalto non sensitive applet	D1390670, A02 Oct 2018	Electronic document via secure file download
Document	Verification process of Third Party non sensitive applet	D1390671, A02 Oct 2018	Electronic document via secure file download
Document	Rules for applications on Multiapp certified product	D1390963, Rev. 1.3 Oct 2018	Electronic document via secure file download
Document	MultiApp Essential v1.1: AGD_PRE document - Javacard Platform	D1462570, Rev 1.8 April 9, 2019	Electronic document via secure file download
Document	MultiApp Essential v1.1: AGD_OPE document - Javacard Platform	D1462568, Rev 1.4 March 8, 2019	Electronic document via secure file download
The following item should be provided only in case of specific commercial agreement and through NDA.			
Electronic file (html files contained on zip file)	D1484942_GTO API_MAE11_PLTF.7z	D1484942, Rev1.0 19/12/2018	Electronic document via secure file download

2.5 LIFE-CYCLE

2.5.1 Product Life-cycle

2.5.1.1 Actors

Actors	Identification
Integrated Circuit (IC) Developer	INVIA
Embedded Software Developer	Gemalto
Integrated Circuit (IC) Manufacturer	INVIA
Module Manufacturer	Gemalto
Card manufacturer (Initializer/Pre-personalizer)	Gemalto
Personalization Agent (Personalizer)	The agent who is acting on the behalf of the Issuer (e.g. issuing State or Organization) and personalize the TOE and applicative data (e.g. MRTD for the holder) by activities establishing the identity of the user (e.g. holder with biographic data).
Card Holder	The rightful holder of the card for whom the issuer personalizes it.

Table 4: Identification of the actors

2.5.1.2 *Life Cycle*

Phase	Description / comments		Who	Where
1	MultiApp Essential v1.1 platform development	Platform development & tests	**Gemalto** GP R&D team - secure environment -	**Gemalto Singapore**
			Gemalto SL Crypto team - secure environment -	**Gemalto Meudon**
	IAS applet development	- Applet development - Applet tests	**Gemalto** GP R&D team - secure environment -	**Gemalto Singapore**
	PSE team	- Script development	**Singapore** **PSE team**	**Gemalto Singapore**
2	IC development	IC development	**IC developer** - Secure environment -	IC development site(s)
3	IC manufacturing	Manufacturing of virgin integrated circuits embedding a flash loader protected by a dedicated transport key.	**IC manufacturer** - Secure environment -	IC manufacturing site(s)
4	SC manufacturing: IC packaging, also called “assembly”	IC packaging & testing	**Gemalto** Production teams - Secure environment -	**Gemalto Gémenos** **Gemalto Singapore**
5	SC manufacturing & pre-personalization	▪ Module embedding in plastic card bodies ▪ Loading of the Gemalto software (platform and applets) ▪ SC initialization (profile building, loading of data needed for card pre-personalization…)	**Gemalto** Production teams - Secure environment -	**Gemalto Gémenos** **Gemalto Tczew** **Gemalto Singapore** **Gemalto Curitiba** **Gemalto Montgomeryville**
6	SC Personalization	Creation of files and loading of end-user data	**SC Personalizer:** Gemalto or another accredited company - Secure environment -	SC Personalizer site
7	End-usage	End-usage for SC issuer	**SC Issuer**	Field
		End-usage for cardholder	**Cardholder**	Field

Figure 4: Life Cycle Table

Additionally, there are sites used for IT support functions as described here below:

Site	IT support activity
Gemalto ATOS Aubervilliers	Configuration Management system hosting servers
Gemalto ATOS Pune	IT Admin of ATOS sites
Gemalto Gémenos	Firewall for office, production and RD networks
Gemalto Calamba	Firewall for office, production and RD networks (backup site)
Gemalto La Ciotat	Applicative Administration of Configuration Management applications

2.5.2 TOE Life-cycle

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

The Java Card System (i.e. the TOE) life-cycle itself can be decomposed in four stages:

- Development
- Storage, pre-personalization and testing
- Personalization and testing
- Final usage

The JCS storage is not necessarily a single step in the life cycle since it can be stored in parts. The JCS delivery occurs before storage and may take place more than once if the TOE is delivered in parts.

These four stages map to the product life cycle phases as shown in Figure 5.

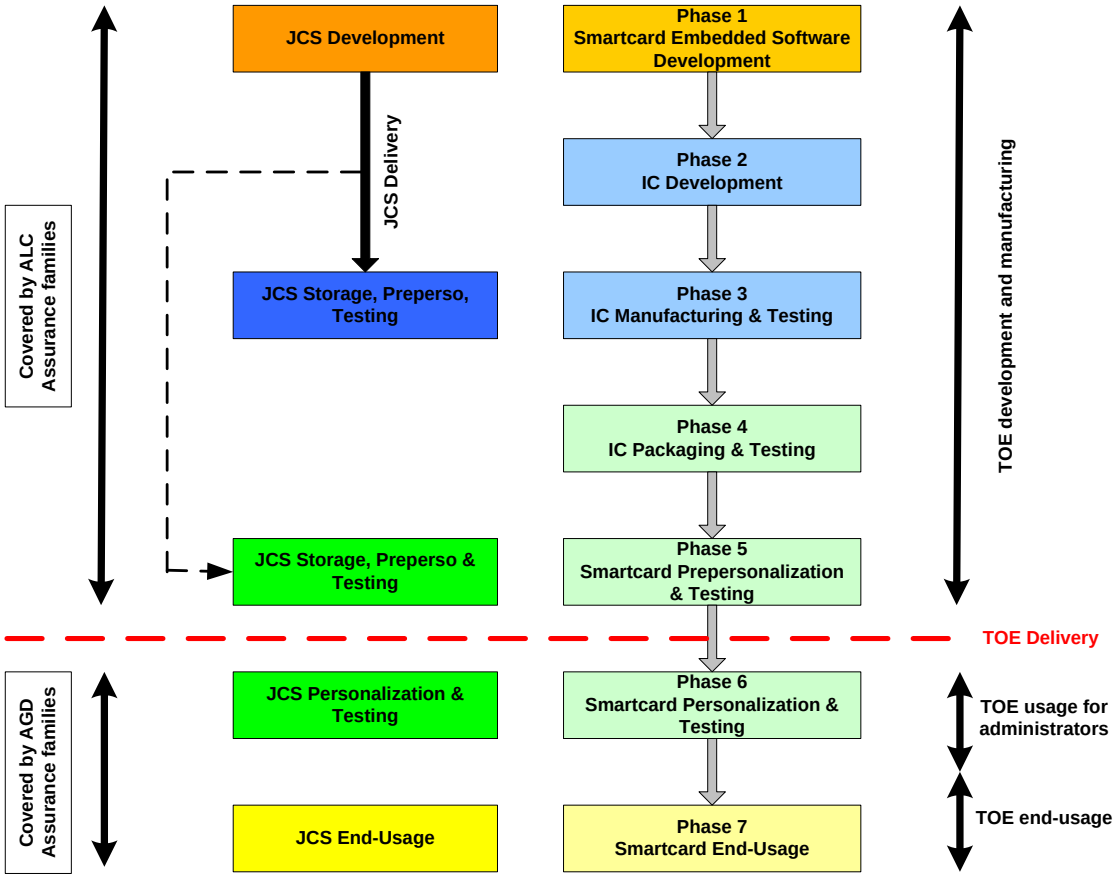

Figure 5: JCS (TOE) Life Cycle within Product Life Cycle

JCS Development is performed during Phase 1. This includes JCS conception, design, implementation, testing and documentation. The JCS development shall fulfill requirements of the final product, including conformance to Java Card Specifications, and recommendations of the SCP user guidance. The JCS development shall occur in a controlled environment that avoids disclosure of source code, data and any critical documentation and that guarantees the integrity of these elements. The present evaluation includes the JCS development environment.

In Phase 3, the IC Manufacturer may store, initialize the JCS and potentially conduct tests on behalf of the JCS developer. The IC Manufacturing environment shall protect the integrity and confidentiality of the JCS and of any related material, for instance test suites. The present evaluation includes the whole IC Manufacturing environment, in particular those locations where the JCS is accessible for installation or testing. As the Security IC has already been certified against [PP-IC-0084] there is no need to perform the evaluation again.

In Phase 5, the SC Pre-Personalizer may store, pre-personalize the JCS and potentially conduct tests on behalf of the JCS developer. The SC Pre-Personalization environment shall protect the integrity and confidentiality of the JCS and of any related material, for instance test suites.

(Part of) JCS storage in Phase 5 implies a TOE delivery after Phase 5. Hence, the present evaluation includes the SC Pre-Personalization environment. The TOE delivery point is placed at the end of Phase 5, since the entire TOE is then built and embedded in the Security IC.

The JCS is personalized in Phase 6, if necessary. The SC Personalization environment is not included in the present evaluation. Appropriate security recommendations are provided to the SC Personalizer through the [AGD] documentation.

The JCS final usage environment is that of the product where the JCS is embedded in. It covers a wide spectrum of situations that cannot be covered by evaluations. The JCS and the product shall provide the full set of security functionalities to avoid abuse of the product by untrusted entities.

Note: Potential applications loaded in pre-issuance will be verified using dedicated evaluated verification process. Applications loaded in post-issuance will need to follow dedicated development rules.

2.5.3 GP Life-cycle

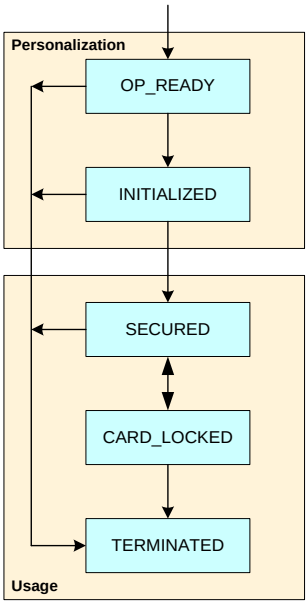

Figure 6: GP Life Cycle

2.6 TOE INTENDED USAGE

2.6.1.1 Personalization Phase

During the Personalization Phase the following Administrative Services are available:

- Applet Load
- Applet Install
- Applet Delete
- Applet Extradite
- Applet Management Lock

All applet management operations require the authentication of the Issuer. By erasing the authentication keys with random numbers, the Issuer can prevent all subsequent applet management operations. This operation is not reversible.
In the Personalization phase, Applet Management Lock is optional.

2.6.1.2 Usage Phase

During the Usage Phase, if the Applet Management lock has not been put, the Administrative Services are available as during the Personalization phase:

- Applet Load
- Applet Install
- Applet Delete
- Applet Extradite
- Applet Management Lock

In addition, the following User services are available:

- Applet Selection
- Applet Interface

2.7 NON-TOE HW / SW / FW AVAILABLE TO THE TOE

The following section mentions the components involved in the environment of the Java Card System which are not part of the TOE

Bytecode Verifier
The bytecode verifier is a program that performs static checks on the bytecodes of the methods of a CAP file prior to the execution of the file on the card. Bytecode verification is a key component of security: applet isolation, for instance, depends on the file satisfying the properties a verifier checks to hold. A method of a CAP file that has been verified shall not contain, for instance, an instruction that allows forging a memory address or an instruction that makes improper use of a return address as if it were an object reference. In other words, bytecodes are verified to hold up to the intended use to which they are defined. Bytecode verification could be performed totally or partially dynamically. No standard procedure in that concern has yet been recognized. Furthermore, different approaches have been proposed for the implementation of bytecode verifiers, most notably data flow analysis, model checking and lightweight bytecode verification, this latter being an instance of what is known as proof carrying code. The actual set of checks performed by the verifier is implementation-dependent, but it is required that it should at least enforce all the “must clauses” imposed in [JCVM304] on the bytecodes and the correctness of the CAP files’ format.

Converter
The converter converts them into Java Card executable code format converted applet (CAP) files so that the bytecode can be downloaded on the card

CAP file loader
The CAP file loader loads the files from the host system to the card, has both an off-card part and an on-card part. This loader is very different from the standard Java class loader in the sense that it is not used to dynamically load classes during the execution of a program but to statically load entire packages on the card.

3 CONFORMANCE CLAIMS

3.1 CC CONFORMANCE CLAIM

Common criteria Version:

This ST conforms to CC Version 3.1 [CC-1] [CC-2] [CC-3]

Conformance to CC part 2 and 3:

- CC part 2 extended with the FCS_RND component. All the other security requirements have been drawn from the catalogue of requirements in Part 2 [CC-2].
- CC part 3 conformant

The Common Methodology for Information Technology Security Evaluation, Evaluation Methodology; [CEM] has to be taken into account.

3.2 PP CLAIM

The MultiApp Essential v1.1: JCS Security Target claims demonstrable conformance to the Protection Profile “JavaCard System – Open configuration”, ([PP-JCS-Open]).

The MultiApp Essential v1.1: JCS Security Target is a composite security target, including the IC security target [ST-IC]. However the security problem definition, the objectives, and the SFR of the IC are not described in this document.

3.3 PACKAGE CLAIM

This ST is conforming to assurance package EAL5 augmented with ALC_DVS.2 and AVA_VAN.5 defined in CC part 3 [CC-3].

3.4 CONFORMANCE STATEMENT

This ST claims demonstrable conformance to [PP-JCS-Open]. The conformance is explained in the rationale.

4 SECURITY ASPECTS

This chapter describes the main security issues of the Java Card System and its environment addressed in this ST, 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”.

4.1 CONFIDENTIALITY

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

4.2 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.

4.3 UNAUTHORIZED EXECUTIONS

#.EXE-APPLI-CODE	Application (byte)code must be protected against unauthorized execution. This concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language ([JAVASPEC]§6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code.;
#.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, 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 the TSFs, is submitted to the same requirements. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

4.4 BYTECODE VERIFICATION

#.VERIFICATION	All bytecode must be verified prior to being executed. Bytecode verification includes (1) how well-formed CAP file is and the verification of the typing constraints on the bytecode, (2) binary compatibility with installed CAP 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.

4.4.1 CAP file Verification

Bytecode verification includes checking at least the following properties: (1) bytecode instructions represent a legal set of instructions used on the Java Card platform; (2) adequacy of bytecode operands to bytecode semantics; (3) absence of operand stack overflow/underflow; (4) control flow confinement to the current method (that is, no control jumps to outside the method); (5) absence of illegal data conversion and reference forging; (6) enforcement of the private/public access modifiers for class and class members; (7) validity of any kind of reference used in the bytecodes (that is, any pointer to a bytecode, class, method, object, local variable, etc actually points to the beginning of piece of data of the expected kind); (8) enforcement of rules for binary compatibility (full details are given in [JCVM304], [JVM], [JCBV]). The actual set of checks performed by the verifier is implementation-dependent, but shall at least enforce all the “must clauses” imposed in [JCVM304] on the bytecodes and the correctness of the CAP files’ format.
As most of the actual Java Card VMs do not perform all the required checks at runtime, mainly because smart cards lack memory and CPU resources, CAP file verification prior to execution is mandatory. On the other hand, there is no requirement on the precise moment when the verification shall actually take place, as far as it can be ensured that the verified file is not modified thereafter. Therefore, the bytecodes can be verified either before the loading of the file on to the card or before the installation of the file in the card or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. This Security Target 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 [JCVM304] §4.4), second, that the export files used to check and link the loaded applet have the corresponding correct counterpart on the card.

4.4.2 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 bytecodes to be verified dynamically. On-card bytecode verifier is out of the scope of this Security Target.

4.4.3 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.

4.5 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 should the installation of a package or an applet fail 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). 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).

4.6 SERVICES

#.ALARM

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

#.OPERATE

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

#.RESOURCES

The TOE controls the availability of resources for the applications and enforces quotas and limitations in order to prevent unauthorized denial of service or malfunction of the TSFs. This concerns both execution (dynamic memory allocation) and installation (static memory allocation) of applications and packages.

#.CIPHER

The TOE shall provide a means to the applications for ciphering sensitive data, for instance, through a programming interface to low-level, highly secure cryptographic services. In particular, those services must support cryptographic algorithms consistent with cryptographic usage policies and standards.

#.KEY-MNGT

The TOE shall provide a means to securely manage cryptographic keys. This includes: (1) Keys shall be generated in accordance with specified cryptographic key generation algorithms and specified cryptographic key sizes, (2) Keys must be distributed in accordance with specified cryptographic key distribution methods, (3) Keys must be initialized before being used, (4) Keys shall be destroyed in accordance with specified cryptographic key destruction methods.

#.PIN-MNGT	The TOE shall provide a means to securely manage PIN objects. This includes: (1) Atomic update of PIN value and try counter, (2) No rollback on the PIN-checking function, (3) Keeping the PIN value (once initialized) secret (for instance, no clear-PIN-reading function), (4) Enhanced protection of PIN's security attributes (state, try counter…) in confidentiality and integrity.
#.SCP	The smart card platform must be secure with respect to the SFRs. Then: (1) After a power loss, RF signal loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state. (2) It does not allow the SFRs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the Java Card API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System. (3) It provides secure low-level cryptographic processing to the Java Card System. (4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism. (5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low–level control accesses (segmentation fault detection). (6) It safely transmits low–level exceptions to the TOE (arithmetic exceptions, checksum errors), when applicable. Finally, it is required that (7) the IC is designed in accordance with a well defined set of policies and standards (for instance, those specified in [PP-IC-0084]), 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).

5 SECURITY PROBLEM DEFINITION

5.1 ASSETS

The assets of the TOE are those defined in [PP-JCS-Open]. The assets of [PP-IC-0084] are studied in [ST-IC].

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.

5.1.1 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.

5.1.2 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 monopolization and unauthorized disclosure or modification.

D.SEC_DATA

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

5.2 THREATS

This section introduces the threats to the assets against which specific protection within the TOE or its environment is required. The threats are classified in several groups.

5.2.1 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**.

5.2.2 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_KEYs**.

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.

5.2.3 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).

5.2.4 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.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**.

5.2.5 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**.

5.2.6 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).

5.2.7 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**.

5.2.8 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.

5.3 ORGANIZATIONAL SECURITY POLICIES

5.3.1 Java Card System Protection Profile – Open Configuration

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

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.
If the application development guidance provided by the platform developer contains recommandations related to the isolation property of the platform, this policy shall also ensure that the verification authority checks that these recommendations are applied in the application code.

5.3.2 TOE additional OSP

OSP.SpecificAPI

The TOE must contribute to ensure that application can optimize control on its sensitive operations using a dedicated API provided by TOE. TOE will provide services for secure array management and to detect loss of data integrity and inconsistent execution flow and react against tearing or fault induction.

OSP.RND

This policy shall ensure the entropy of the random numbers provided by the TOE to applet using [JCAPI304] is sufficient. Thus attacker is not able to predict or obtain information on generated numbers.

5.4 ASSUMPTIONS

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

A.APPLET

Applets loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM304], §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.

5.5 COMPATIBILITY BETWEEN SECURITY ENVIRONMENTS OF [ST-JCS] AND [ST-IC]

5.5.1 Compatibility between threats of [ST-JCS] and [ST-IC]

T.CONFID-JCS-CODE, T.CONFID-APPLI-DATA, and T.CONFID-JCS-DATA are included in T.Phys-Probing, T.Leak-Inherent and T.Leak-Forced.

T.SID.2 is partly included in T.Phys-Manipulation and T.Malfunction.

T.PHYSICAL is included in T.Phys-Probing, T.Leak-Inherent, T.Phys-Manipulation, T.Malfunction and T.Leak-Forced.

T.INTEG-APPLI-CODE T.INTEG-JCS-CODE T.INTEG-APPLI-DATA DATA T.INTEG-JCS-DATA T.INTEG-APPLI-CODE.LOAD T.INTEG-APPLI-DATA.LOAD T.SID.1 T.EXE-CODE.1 T.EXE-CODE.2 T.NATIVE T.RESOURCES T.INSTALL T.DELETION T.OBJ-DELETION are threats specific to the Java Card platform and they do no conflict with the threats of [ST-IC].

5.5.2 Compatibility between OSP of [ST-JCS] and [ST-IC]

OSP.VERIFICATION is an OSP specific to the Java Card platform and it does no conflict with the OSP of [ST-IC].
OSP.SpecificAPI has been added to this [ST-JCS] in order to manage Specific API and it does no conflict with the OSP of [ST-IC].
OSP.RND has been added to this [ST-JCS] in order to manage RNG and it does no conflict with the OSP of [ST-IC].

We can therefore conclude that the OSP for the environment of [ST-JCS] and [ST-IC] are consistent.

5.5.3 Compatibility between assumptions of [ST-JCS] and [ST-IC]

A.VERIFICATION, A.APPLET are assumptions specific to the Java Card platform and they do no conflict with the assumptions of [ST-IC].
We can therefore conclude that the assumptions for the environment of [ST-JCS] and [ST-IC] are consistent.

6 SECURITY OBJECTIVES

6.1 SECURITY OBJECTIVES FOR THE TOE

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

6.1.1 Identification

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

6.1.2 Execution

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

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

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

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

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

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

Application note:
To be made unavailable means to be physically erased with a default value. Except for local variables that do not correspond to method parameters, the default values to be used are specified in [JCVM304].

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

6.1.3 Services

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

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

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

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

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

Application note:
O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION and O.CIPHER are actually provided to applets in the form of Java Card APIs. Vendor-specific libraries 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. Depending on whether they contain native code or not, these proprietary libraries will need to be evaluated together with the TOE or not (see #.NATIVE). In any case, they are not included in the Java Card System for the purpose of the present document.

6.1.4 Object deletion

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

6.1.5 Applet management

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

O.LOAD
The TOE shall ensure that the loading of a package into the card is safe.
Besides, for codes loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the verification authority. This verification by the TOE shall occur during the load or late during the install process.

O.INSTALL
The TOE shall ensure that the installation of an applet performs as expected. (See #.INSTALL for details).
Besides, for codes loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the verification authority. If not performed during the loading process, this verification by the TOE shall occur during the install process.

6.1.6 SCP

The Objectives described in this section are Objectives for the Environment in [PP-JCS-Open]. They become Objectives for the TOE because the TOE in this ST includes the SCP.

O.SCP.RECOVERY
If there is a loss of power, or if the smart card is withdrawn from the CAD while an operation is in progress, the SCP must allow the TOE to eventually complete the interrupted operation successfully, or recover to a consistent and secure state.
This security objective for the TOE refers to the security aspect #.SCP.1: The smart card platform must be secure with respect to the SFRs. Then after a power loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state.

O.SCP.SUPPORT
The SCP shall support the TSFs of the TOE.
This security objective for TOE refers to the security aspect #.SCP.2-5:
(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).

O.SCP.IC
The SCP 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.

6.1.7 CMGR

The Objectives described in this section are Objectives for the Environment in [PP-JCS-Open]. They become Objectives for the TOE because the TOE in this ST includes the Card Manager.

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

6.1.8 Additional objectives

Objectives described in this section are additional objectives related to the TOE.

O.SpecificAPI
The TOE shall provide to application a specific API means to optimize control on sensitive operations performed by application.
TOE shall provide services for secure array management and to detect loss of data integrity and inconsistent execution flow and react against tearing or fault induction.

O.RND
The TOE must contribute to ensure that random numbers shall not be predictable and shall have sufficient entropy.

6.2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

6.2.1 Objectives extracted from [PP-JCS-Open]

This section introduces the security objectives to be achieved by the environment and extracted from [PP-JCS-Open].

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

Additionally, the applet shall follow all recommendations, if any, mandated in the platform guidance for maintaining the isolation property of the platform.
Application Note:
Constraints to maintain the isolation property of the platform are provided by the platform developer in application development guidance. The constraints apply to all application code loaded in the platform.

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

OE.CODE-EVIDENCE
For application code loaded pre-issuance, evaluated technical measures implemented by the TOE or audited organizational measures must ensure that loaded application has not been changed since the code verifications required in OE.VERIFICATION.
For application code loaded post-issuance and verified off-card according to the requirements of OE.VERIFICATION, the verification authority shall provide digital evidence to the TOE that the application code has not been modified after the code verification and that he is the actor who performed code verification.
For application code loaded post-issuance and partially or entirely verified on-card, technical measures must ensure that the verification required in OE.VERIFICATION are performed. On-card bytecode verifier is out of the scope of this Protection Profile.
Application Note:
For application code loaded post-issuance and verified off-card, the integrity and authenticity evidence can be achieved by electronic signature of the application code, after code verification, by the actor who performed verification.

6.3 SECURITY OBJECTIVES RATIONALE

	O.SID	O.OPERATE	O.RESOURCES	O.FIREWALL	O.NATIVE	O.REALLOCATION	O.GLOBAL_ARRAYS_CONFID	O.GLOBAL_ARRAYS_INTEG	O.ALARM	O.TRANSACTION	O.CIPHER	O.PIN_MNGT	O.KEY_MNGT	O.OBJ_DELETION	O.INSTALL	O.LOAD	O.DELETION	O.SCP.RECOVERY	O.SCP.SUPPORT	O.SCP.IC	O.CARD_MANAGEMENT	O.SpecificAPI	O.RND	OE.VERIFICATION	OE.APPLET	OE;CODE_EVIDENCE
T.CONFID-JCS-CODE					X																X			X		
T.CONFID-APPLI-DATA	X	X		X		X	X		X	X	X	X	X					X	X		X			X		
T.CONFID-JCS-DATA	X	X		X					X									X	X		X			X		
T.INTEG-APPLI-CODE					X																X			X		X
T.INTEG-JCS-CODE					X																X			X		X
T.INTEG-APPLI-DATA	X	X		X		X		X	X	X	X	X	X					X	X		X			X		X
T.INTEG-JCS-DATA	X	X		X					X									X	X		X			X		X
T.INTEG-APPLI-CODE.LOAD																X					X					X
T.INTEG-APPLI-DATA.LOAD																X					X					X
T.SID.1	X			X			X	X							X						X					
T.SID.2	X	X		X											X			X	X							
T.EXE-CODE.1				X																				X		
T.EXE-CODE.2																								X		
T.NATIVE					X																			X	X	
T.RESOURCES		X	X												X			X	X							
T.INSTALL															X	X					X					
T.DELETION																	X				X					
T.OBJ-DELETION														X												
T.PHYSICAL																				X						
OSP.VERIFICATION																X								X		X
OSP.SpecificAPI																						X				
OSP.RND																							X			
A.APPLET																									X	
A.VERIFICATION																								X		X

6.3.1 Threats

6.3.1.1 Confidentiality

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-APPLI-DATA This threat is countered by the security objective for the operational environment regarding bytecode verification (OE.VERIFICATION). It is also covered by the isolation commitments stated in the (O.FIREWALL) objective. It relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives 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 (O.GLOBAL_ARRAYS_CONFID) security objective.
Finally, any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the O.REALLOCATION objective. That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused.

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

6.3.1.2 Integrity

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

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 disclose or modify a piece of code.
The (#.VERIFICATION) security aspect is addressed in this configuration by the objective for the environment OE.VERIFICATION.
The objectives O.CARD_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity.
T.INTEG-APPLI-DATA This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives O.CARD_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.
Concerning the confidentiality and integrity of application sensitive data, as applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys and PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) is also concerned.
Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The integrity of the information stored in that buffer is ensured by the (O.GLOBAL_ARRAYS_INTEG) objective.
Finally, any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the O.REALLOCATION objective. That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused.

T.INTEG-JCS-DATA This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives O.CARD_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity.
The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

T.INTEG-APPLI-CODE.LOAD This threat is countered by the security objective O.LOAD which ensures that the loading of packages is done securely and thus preserves the integrity of packages code.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and

authenticity. By controlling the access to card management functions such as the installation, update or deletion of applets the objective O.CARD_MANAGEMENT contributes to cover this threat.

T.INTEG-APPLI-DATA.LOAD This threat is countered by the security objective O.LOAD which ensures that the loading of packages is done securely and thus preserves the integrity of applications data.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. By controlling the access to card management functions such as the installation, update or deletion of applets the objective O.CARD_MANAGEMENT contributes to cover this threat.

6.3.1.3 Identity usurpation

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

T.SID.2 This is covered by integrity of TSF data, subject-identification (O.SID), the firewall (O.FIREWALL) and its good working order (O.OPERATE).
The objective O.INSTALL contributes to counter this threat by ensuring that installing an applet has no effect on the state of other applets and thus can't change the TOE's attribution of privileged roles.
The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE objective of the TOE, so they are indirectly related to the threats that this latter objective contributes to counter.

6.3.1.4 Unauthorized execution

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

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

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

6.3.1.5 Denial of service

T.RESOURCES This threat is directly countered by objectives on resource-management (O.RESOURCES) for runtime purposes and good working order (O.OPERATE) in a general manner.
Consumption of resources during installation and other card management operations are covered, in case of failure, by O.INSTALL.
It should be noticed that, for what relates to CPU usage, the Java Card platform is single-threaded and it is possible for an ill-formed application (either native or not) to monopolize the CPU. However, a smart card can be physically interrupted (card removal or hardware reset) and most CADs implement a timeout policy that prevent them from being blocked should a card fails to answer. That point is out of scope of this Security Target, though.

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

6.3.1.6 Card management

T.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.

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.

6.3.1.7 Services

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

6.3.1.8 Miscellaneous

T.PHYSICAL Covered by O.SCP.IC. Physical protections rely on the underlying platform.

6.3.2 Organizational Security Policies

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

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

This policy is also upheld by O.LOAD ensures that the loading of a package into the card is safe, Besides, ensures that the TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the verification authority occur during the load or late during the install process.

OSP.SpecificAPI This OSP is enforced by the TOE security objective O.SpecificAPI.

OSP.RND This OSP is enforced by the TOE security objective O.RND.

6.3.3 Assumptions

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

A.VERIFICATION This assumption is upheld by the security objective on the operational environment OE.VERIFICATION which guarantees that all the bytecodes shall be verified at least once, before the loading, before the installation or before the execution in order to ensure that each bytecode is valid at execution time. This assumption is also upheld by the security objective of the environment OE.CODE-EVIDENCE which ensures that evidences exist that the application code has been verified and not changed after verification.

6.3.4 Compatibility between objectives of [ST-JCS] and [ST-IC]

6.3.4.1 Compatibility between objectives for the TOE

O.SID, O.OPERATE, O.RESOURCES, O.FIREWALL, O.NATIVE, O.REALLOCATION, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.ALARM; O.TRANSACTION, O.PIN-MNGT, O.KEY-MNGT, O.OBJ-DELETION, O.INSTALL, O.LOAD, O.DELETION, O.CARD-MANAGEMENT, and O.SCP.RECOVERY are objectives specific to the Java Card platform and they do no conflict with the objectives of [ST-IC].
O.SpecificAPI is objective added to this platform it does no conflict with the objectives of [ST-IC].
O.RND added to this platform is related with the following objectives of [ST-IC]: O.RND
O.CIPHER is related with the following objectives of [ST-IC]: O.RND.
O.SCP.SUPPORT is partially related with the following objectives of [ST-IC]: O.RND.
O.SCP.IC is related with the following objectives of [ST-IC]: O.Phys-Manipulation, O.Phys-Probing, O.Malfunction, O.Leak-Inherent, O.Leak-Forced and O.Abuse-Func.

We can therefore conclude that the objectives for the TOE of [ST-JCS] and [ST-IC] are consistent.

6.3.4.2 Compatibility between objectives for the environment

OE.VERIFICATION, OE.CODE-EVIDENCE and OE.Applet are objectives specific to the Java Card platform and they do no conflict with the objectives of [ST-IC].
We can therefore conclude that the objectives for the environment of [ST-JCS] and [ST-IC] are consistent.

7 EXTENDED COMPONENTS DEFINITION

7.1 DEFINITION OF THE FAMILY FCS_RND

To define the IT security functional requirements of the TOE a sensitive family (FCS_RND) of the Class FCS (cryptographic support) is defined here. This family describes the functional requirements for random number generation used for cryptographic purposes. The component FCS_RND is not limited to generation of cryptographic keys unlike the component FCS_CKM.1. The similar component FIA_SOS.2 is intended for non-cryptographic use.

The family “Generation of random numbers (FCS_RND)” is specified as follows.

FCS_RND Generation of random numbers

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

Component levelling:

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

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

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

FCS_RND.1 Quality metric for random numbers

Hierarchical to: No other components
Dependencies: No dependencies

FCS_RND.1.1 The TSF shall provide a mechanism to generate random numbers that meet [assignment: *a defined quality metric*].

8 SECURITY REQUIREMENTS

The operations, assignment, selection, iteration and refinement on the security requirements are marked as follows:

- Assignment: Between brackets and bold, e.g: **[Bold]**
- Selection: Between brackets and with the word selection at the beginning, e.g: [selection: selection choosen]
- Iteration: The name of the SFR follow by a '/' character and the iteration name, e.g: SFRid/Iteration
- Refinement: The word 'refinement' at the beginning of the paragraph in italics, e.g: *Refinement*

8.1 SECURITY FUNCTIONAL REQUIREMENTS

This section states the security functional requirements for the Java Card System – Open configuration.

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

The SFRs refer to all potentially applicable subjects, objects, information, operations and security attributes.

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

Subject	Description
S.ADEL	The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([JCRE304], §11), but its role asks anyway for a specific treatment from the security viewpoint. This subject is unique and is involved in the ADEL security policy defined in §8.1.3.

Subject	Description
S.APPLET	Any applet instance.
S.BCV	The bytecode verifier (BCV), which acts on behalf of the verification authority who is in charge of the bytecode verification of the packages. This subject is involved in the PACKAGE LOADING security policy defined in §8.1.6.
S.CAD	The CAD represents 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.

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.

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

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

Security attribute	Description/Value
Active Applets	The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels.
Applet Selection Status	"Selected" or "Deselected".
Applet's version number	The version number of an applet (package) indicated in the export file.
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 ([JCVM304], §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 a 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

Security attribute	Description/Value
	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 instance registered on the card.
ResidentPackages	The set of AIDs of the packages already loaded on the card.
Selected Applet Context	Package AID or "None".
Sharing	Standards, SIO, Java Card RE entry point or global array.
Static References	Static fields of a package may contain references to objects. The Static References attribute records those references.

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

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

Operation	Description
OP.ARRAY_ACCESS(O.JAVAOBJECT, field)	Read/Write an array component.
OP.CREATE(Sharing, LifeTime) (*)	Creation of an object (new or makeTransient call).
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed applet and its objects, either logically or physically.
OP.DELETE_PCKG(O.CODE_PKG,...)	Delete a package, either logically or physically.
OP.DELETE_PCKG_APPLET(O.CODE_PKG,...)	Delete a package and its installed applets, either logically or physically.
OP.INSTANCE_FIELD(O.JAVAOBJECT, field)	Read/Write a field of an instance of a class in the Java programming language.
OP.INVK_VIRTUAL(O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object).
OP.INVK_INTERFACE(O.JAVAOBJECT, method, arg1,...)	Invoke an interface method.
OP.JAVA(...)	Any access in the sense of [JCRE304], §6.2.8. It stands for one of the operations OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW, OP.TYPE_ACCESS.
OP.PUT(S1,S2,I)	Transfer a piece of information I from S1 to S2.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see [JCRE304],§6.2.8.7)
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	Invoke checkcast or instanceof on an object in order to access to classes (standard or shareable interfaces objects).

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

8.1.1 CoreG_LC Security Functional Requirements

This group is focused on the main security policy of the Java Card System, known as the firewall. This policy essentially concerns the security of installed applets. The policy focuses on the execution of bytecodes.

8.1.1.1 Firewall Policy

FDP_ACC.2/FIREWALL Complete access control

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

Refinement:
The operations involved in the policy are:

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

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

Application note:
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	Attributes
S.PACKAGE	**LC Applet Selection Status**
S.JCVM	**ActiveApplets, 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 ([JCRE304]§6.2.8) An S.PACKAGE may freely perform any of OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW or OP.TYPE_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".**
- **R.JAVA.2 ([JCRE304]§6.2.8) An S.PACKAGE may freely perform any of OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE or OP.THROW upon any 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 ([JCRE304]§6.2.8.10) An 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 ([JCRE304], §6.2.8.6,) An S.PACKAGE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if the invoked interface method extends the Shareable interface and one of the following applies:**

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

  (b) The value of the attribute Selection Status of the package whose AID is "Package AID' is "Non-multiselectable», and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute ActiveApplets.
- **R.JAVA.5 An S.PACKAGE may perform an OP.CREATE only if the value of the Sharing parameter(*) is "Standard".]**

FDP_ACF.1.3/FIREWALL The TSF shall explicitly authorize 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**.]

Application note:

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

• Standard ones, whose both fields and methods are under the firewall policy,

• Shareable interface Objects (SIO), which provide a secure mechanism for inter-applet communication,

• JCRE entry points (Temporary or Permanent), who have freely accessible methods but protected fields,

• Global arrays, having both unprotected fields (including components; refer to 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 ([JCRE304], §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).

([JCRE304], 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.
([JCRE304], §6.1.2.1) At any point in time, there is only one active context within the Java Card VM (this is called the Currently Active Context).

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

The Java Card platform, version 2.2.x 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 ([JCVM304], §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. ([JCRE304], §4).

FDP_IFC.1/JCVM Subset information flow control

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

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

FDP_IFF.1/JCVM Simple security attributes

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

Subject / Information	Description
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 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 **[no additional information flow control SFP rules**.**]**

FDP_IFF.1.4/JCVM The TSF shall explicitly authorize an information flow based on the following rules: **[no additional information flow control SFP rules**.**]**

FDP_IFF.1.5/JCVM The TSF shall explicitly deny an information flow based on the following rules: **[no additional information flow control SFP rules**.**]**

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]**.

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 **[the selected applet Context security attribute]** to **[the Java Card RE (S.JCRE)**.**]**
Application note:
The modification of the Selected Applet Context is performed in accordance with the rules given in [JCRE304], §4 and [JCVM304], §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 **[the currently active context and the Active Applets security attributes]** to **[the Java Card VM (S.JCVM)**.**]**
Application note:
The modification of the Selected Applet Context is performed in accordance with the rules given in [JCRE304], §4 and [JCVM304], §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**.**]**

FMT_MSA.3/FIREWALL Static attribute initialization

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 *Refinement:* 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_MSA.3/JCVM Static attribute initialization

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 *Refinement:* 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_SMR.1/JCRE Security roles

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

- **the Java Card RE (JCRE)**.
- **the Java Card VM (JCVM).]**

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

Application Note:
FMT_SMR.1.1/JCRE such iteration in this ST [ST-JCS] is similar to FMT_SMR.1 in the [PP-JCS-Open]. It has been used to clarify the situation due to several iterations of FMT_SMR.1
In the context of this SFR, the name JCRE covers both JCRE and JCVM.

FMT_SMF.1/CORE_LC Specification of Management Functions

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

- **Modify the Currently Active Context, the Selected Applet Context, and the Active Applets]**

Application Note:
FMT_SMF.1.1/Core_LC such iteration in this ST [ST-JCS] is similar to FMT_SMF.1 in the [PP-JCS-Open]. It has been used to clarify the situation due to several iterations of FMT_SMF.1

8.1.1.2 Application Programming Interface

The following SFRs are related to the Java Card API.
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 the interface or the invocation mechanism.

For this section, a presentation choice has been selected. Each SFR may present a table with different type of algorithms treated. For each case, there is no distinction regarding the technical objectives fulfilled by each row on the table (thus algorithm family). The technical objectives are the same disregarding this differentiation.

FCS_CKM.1 Cryptographic key generation

FCS_CKM.1.1 The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **[Table 5, column no. 2 'Algorithm and usage']** and specified cryptographic key sizes **[Table 5, column no. 3 'Key size']** that meet the following: **[Table 5, column no. 4 'Standard']**.

Iteration	Algorithm and usage	Key size	Standard
/RSA Std	**RSA standard key generation**	1024(*), 1536(*), **2048**	**ANSI X9.31**
/RSA CRT	**RSA CRT key generation**	1024(*), 1536(*), **2048, 3072**	**ANSI X9.31**
/GP	**GP session keys**	**112 (for SCP01, SCP02 (**))** **128,192, 256 (for SCP03)**	**[GP221] (for SCP01, SCP02 (**))** **[GP221] (for SCP03)**
/DH	**DH key generation**	1024(*), 1536(*), **2048**	**ANSI X9.42**

Table 5: FCKM.1 Cryptographic key generation table.

Application note:

- The keys are generated and diversified in accordance with [JCAPI304] specification in classes KeyBuilder and KeyPair (at least Session key generation).
- In addition to the information above, this TOE supports TDES key lengths of 112 bits and 168 bits. Key length of 112 bits is mentioned on GP specifications and key length of 168 bits key length is mentioned in ISO/IEC 18033-3.
- (*) These key sizes are supported by the product but not conformant with SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms [ACM].
- (**) The use of protocols SCP01 and SCP02 must be in accordance with guides: [AGD-Ref] and [Applet guidance-1]. To recall that SCP01 protocol is deprecated.

FCS_CKM.2 Cryptographic key distribution

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **[Table 6, column no. 3 'Distribution method']** that meets the following: **[Table 6, column no. 4 'Standard']**.

For Applications:

	Iteration	Distribution method	Standard
For applications	**/RSA**	**JC API setkey()**	**[JCAPI304]**
	/TDES	**JC API setkey()**	**[JCAPI304]**
	/AES	**JC API setkey()**	**[JCAPI304]**
	/DH	**Gemalto API setkey()**	**Gemalto API**
For GP	**/STORE Data**	**STORE Data command**	**Global Platform [GP]**
	/PUT KEY	**PUT KEY command**	**Global Platform [GP]**

Table 6: FCKM.2.1 Cryptographic key distribution table.

FCS_CKM.3 Cryptographic key access

FCS_CKM.3.1 The TSF shall perform **[key access]** in accordance with a specified cryptographic key access method **[Table 7, column no. 2 'Key access method']** that meets the following: **[Table 7, column no. 3 'Standard']**.

Iteration	Key access method	Standard
/RSA	**JC API getkey()**	**[JCAPI304]**
/TDES	**JC API getkey()**	**[JCAPI304]**
/AES	**JC API getkey()**	**[JCAPI304]**
/DH	**JC API getkey()**	**[JCAPI304]**

Table 7: FCKM.3.1 Cryptographic key access table.

FCS_CKM.4 Cryptographic key destruction

FCS_CKM.4.1 The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **[secure erasing by overwriting keys to a clear value]** that meets the following: **[None]**.
Application note:

- The keys are reset in accordance with [JCAPI304] in class Key with the method clearKey(). Any access to a cleared key attempting to use it for ciphering or signing shall throw an exception.

FCS_COP.1 Cryptographic operation

FCS_COP.1.1 The TSF shall perform **[Table 8, column no. 2 'Operation']** in accordance with a specified cryptographic algorithm **[Table 8, column no. 3 'Algorithm']** and cryptographic key sizes **[Table 8, column no. 4 'Key size']** that meet the following: **[Table 8, column no. 5 'Standards']**.

Iteration	Operation	Algorithm	Key size	Standards
/RSA-SIGN	**signature & verification**	**RSA (STD)** **RSA (CRT)**	**1024(*), 1152(*), 1280(*), 1536(*) and 2048** **3072**	**[ISO9796-2] RSA SHA PKCS#1** **RSA SHA PKCS#1 PSS**
/RSA-CIPHER	**Encryption & decryption**	**RSA (STD)** **RSA (CRT)**	**1024(*), 1152(*), 1280(*), 1536(*) and 2048** **3072**	**[ISO9796-2] RSA SHA PKCS#1** **RSA SHA PKCS#1 PSS**
/TDES-CIPHER	**Encryption & decryption**	**TDES**	**112** **168**	**[SP800-67] [ISO9797-1]** **DES NOPAD** **DES PKCS#5** **DES 9797 M1 M2**
/AES-CIPHER	**Encryption & decryption**	**AES**	**128, 192, 256**	**[FIPS197], AES 128 NOPAD**
/TDES-MAC	**Signature, Verification**	**TDES**	**112, 168**	**[SP800-67] [ISO9797-1]** **DES MAC ISO9797-1 M1 M2, M3** **DES MAC NOPAD** **DES MAC PKCS#5**
/AES-CMAC	**Signature, Verification**	**AES**	**128, 192, 256**	**[FIPS197] AES 128 NOPAD; SP800-38B**
/SHA	**Hashing**	**Hashing**	**SHA-1(**), SHA-224, SHA-256, SHA-384, SHA-512**	**NA**

Table 8: FCS_COP.1 Cryptographic operation table.

Application note:

- (*) These key sizes are supported by the product but not conformant with SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms [ACM].
- (**) SHA-1 is supported by the product must but not conformant with SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms [ACM].

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**.**]**

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.]**

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.]**

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).]**

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.]**

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 **[O.JAVAOBJECT.]**

FDP_ROL.1.2/FIREWALL The TSF shall permit operations to be rolled back within the **[scope of a select(), deselect(), process(), install() or uninstall() call, notwithstanding the restrictions given in [JCRE304], §7.7, within the bounds of the Commit Capacity ([JCRE304], §7.8), and those described in [JCAPI304].]**

8.1.1.3 Card Security Management

FAU_ARP.1 Security alarms

FAU_ARP.1.1 The TSF shall take **[the following actions:**
- **throw an exception,**
- **or lock the card session**
- **or reinitialize the Java Card System and its data**
- **no other list of other actions]**

upon detection of a potential security violation.

Refinement:
The TOE detects the following potential security violation:
- CAP file inconsistency
- Applet life cycle inconsistency
- Card Manager life cycle inconsistency
- Card tearing (unexpected removal of the Card out of the CAD) and power failure
- Abortion of a transaction in an unexpected context (see abortTransaction(), [JCAPI304] and ([JCRE304], §7.6.2)
- Violation of the Firewall or JCVM SFPs
- Unavailability of resources

- Array overflow
- **[Random trap detection]**

FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for **[integrity errors]** on all objects, based on the following attributes: **[integrity-sensitive data]**.

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **[**

- **Prevent the use of modified data**
- **Raise an exception]**

FPR_UNO.1 Unobservability

FPR_UNO.1.1 The TSF shall ensure that **[unauthorized users]** are unable to observe the operation **[cryptographic operations / comparisons operations]** on **[Key values / PIN values]** by **[S.JCRE, S.Applet.]**

FPT_FLS.1/JCS Failure with preservation of secure state

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

Application note:

- FPT_FLS.1/JCS such iteration in this ST [ST-JCS] is similar to FPT_FLS.1 in the [PP-JCS-Open]. It has been used to clarify the situation due to several iterations of FPT_FLS.1.
- The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE304], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE304]). Behavior of the TOE on power loss and reset is described in [JCRE304], §3.6, and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE304], §3.6.2.

FPT_TDC.1 Inter-TSF basic TSF data consistency

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

FPT_TDC.1.2 The TSF shall use **[**

- **The rules defined in [JCVM304] specification;**
- **The API tokens defined in the export files of reference implementation**
- **The rules defined in ISO 7816-6**
- **The rules defined in [GP221] specification]**

when interpreting the TSF data from another trusted IT product.

8.1.1.4 AID Management

FIA_ATD.1/AID User attribute definition

FIA_ATD.1.1/AID The TSF shall maintain the following list of security attributes belonging to individual users:
[
- **package AID**
- **Applet's version number**
- **registered applet's AID**
- **applet selection status ([JCVM304], §6.5)**.

]
Application note:
- "Individual users" stands for applets.

FIA_UID.2/AID User identification before any action

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

Application notes:
- 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/JCRE is attached to an IT security function rather than to a "user" of the CC terminology. The Java Card RE does not "identify" itself with respect to the TOE, but it is a 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: **[**

- **Initial applet selection is performed as described in [JCRE304]§4**
- **The default applet depends on personalization.]**

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: **[**

- **Applet selection is performed after a successful SELECT FILE command as described in [JCRE304]§4.]**

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]**.

FMT_MTD.3/JCRE Secure TSF data

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

8.1.2 INSTG Security Functional Requirements

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

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.

Application note:

- The format of the CAP file is precisely defined in Sun's specification ([JCVM304]); it contains the user data (like applet's code and data) and the security attribute altogether. Therefore there is no association to be carried out elsewhere.

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.
Application note:

- Each package contains a package Version attribute, which is a pair of major and minor version numbers ([JCVM304], §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 ([JCVM304], §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 compatibles. However, package files do have "package Version Numbers" ([JCVM304]) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.

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 ([JCVM304], §4.5.2) .]

FMT_SMR.1/Installer Security roles

FMT_SMR.1.1/Installer The TSF shall maintain the roles**: [the 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 [JCRE304] §11.1.4.]**

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.
Application note:

- The TOE has no maintenance mode.

FPT_RCV.3.2/Installer For **[Failure during applet loading, installation and deletion; sensitive data loading]**, 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 **[none]** 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.

8.1.3 ADELG Security Functional Requirements

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

FDP_ACC.2/ADEL Complete access control

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

Refinement:
The operations involved in the policy are:
- OP.DELETE_APPLET,
- OP.DELETE_PCKG,
- OP.DELETE_PCKG_APPLET.

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

FDP_ACF.1/ADEL Security attribute based access control

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

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

]

FDP_ACF.1.2/ADEL The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:**[**
In the context of this policy, an object O is reachable if and only if one of the following conditions holds:

(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), and
(4) there exists an object O' that is reachable according to either (1) or (2) or (3) above and O' contains a reference to O (the reachability status of O is that of O').

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

R.JAVA.14 ([JCRE304], §11.3.4.1, Applet Instance Deletion). The S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if,

(1) S.ADEL is currently selected,
(2) 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 ([JCRE304], §8.5) O.JAVAOBJECT is remote reachable.

R.JAVA.15 ([JCRE304], §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 in the context of O.APPLET 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 ([JCRE304], §8.5) O.JAVAOBJECT is remote reachable.

R.JAVA.16 ([JCRE304], §11.3.4.2, Applet/Library Package Deletion). The S.ADEL may perform OP.DELETE_PCKG upon an O.CODE_PCKG only if,

(1) S.ADEL is currently selected,

(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PCKG that is an instance of a class that belongs to O.CODE_PCKG exists on the card and

(3) there is no resident package on the card that depends on O.CODE_PCKG.

R.JAVA.17 ([JCRE304], §11.3.4.3, Applet Package and Contained Instances Deletion). S.ADEL may perform OP.DELETE_PCKG_APPLET upon an O.CODE_PCKG only if,

(1) S.ADEL is currently selected,

(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PCKG, which is an instance of a class that belongs to O.CODE_PCKG exists on the card,

(3) there is no package loaded on the card that depends on O.CODE_PCKG and

(4) for every O.APPLET of those being deleted it holds that:

(i) 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 ([JCRE304],§8.5) O.JAVAOBJECT is remote reachable.

]

Application notes:

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

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

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

FDP_RIP.1/ADEL Subset residual information protection

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

Application note:

- Requirements on de-allocation during applet/package deletion are described in [JCRE304], §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 (S.JCRE).]**
Application note:
The modification of the ActiveApplets security attribute should be performed in accordance with the rules given in [JCRE304], §4.

FMT_MSA.3/ADEL Static attribute initialization

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: **[the 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 [JCRE304], §11.3.4.]**
Application note:

- The applet instance deletion must be atomic. The "secure state" referred to in the requirement must comply with the Java Card specifications. That is, if a reset or power fail occurs during the deletion process, then before any applet is selected in card, either the applet instance deletion is completed or the applet shall be selectable and all objects owned by the applet remain unchanged (that is, the functionality of all applet instances on the card remains the same as prior to the unsuccessful deletion attempt) [JCRE304], §11.3.4.

8.1.4 ODELG Security Functional Requirements

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

FDP_RIP.1/ODEL Subset residual information protection

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

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**.**]**

8.1.5 CarG Security Functional Requirements

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

FCO_NRO.2/CM Enforced proof of origin

FCO_NRO.2.1/CM The TSF shall enforce the generation of evidence of origin for transmitted **[application packages]** at all times.
Application note:
Upon reception of a new application package for installation, the card manager shall first check that it actually comes from the verification authority. The verification authority is the entity responsible for bytecode verification.

FCO_NRO.2.2/CM *Refinement:* 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 **[no limitation]**.

FDP_IFC.2/CM Complete information flow control

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

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

Application note:

The subjects covered by this policy are those involved in the loading of an application package by the card through a potentially unsafe communication channel:

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

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:
[

Subject / Information	Attribute	value
User	**role**	**Operator, Owner, Issuer, None**
Applet	**checked**	**Boolean**
DAP Key	**OK**	**Boolean**

]
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: **[**

- **The user with the security attribute role set to Operator or Issuer can load an applet.**
- **Only applets with the security attribute Checked set to YES can be transferred.**

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

FDP_IFF.1.4/CM The TSF shall explicitly authorize an information flow based on the following rules: **[**

- **The Issuer, behaving as the BCV, can load it through a secure channel, after having verified the applet.**
- **The Issuer can load an applet with a DAP specifying that it has been verified by the BCV.**
- **The Operator, having checked the applet can load it through a secure channel.**

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

- **The TOE fails to verify the integrity and authenticity evidences of the application package**
- **An applet, not verified by a BCV cannot be loaded.]**

FDP_UIT.1/CM Data exchange integrity

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

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

Application note:

Modification errors 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.

The following SFR is an iteration of FIA_UAU.1 which is taken from [CC-2].

FIA_UAU.1/CM Timing of authentication

FIA_UAU.1.1/CM The TSF shall allow **[**
- **To use JCAPI with already installed applets**
- **To send APDUs for Applets]**

on behalf of the user to be performed before the user is authenticated.

Application note:

This authentication of the card manager is a strong authentication as soon as the TOE leaves the protected environment of audited facilities. For this purpose, keys are diversified.

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

FIA_UID.1/CM Timing of identification

FIA_UID.1.1/CM The TSF shall allow **[**
- **To use JCAPI with already installed applets**
- **To send APDUs for Applets]**

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.

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 [selection:modify] the security attributes **[AID]** to **[None]**.

FMT_MSA.3/CM Static attribute initialization

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 **[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:

- **[The loading of the applets, with their AID by the Card Manager]**.

FMT_SMR.1/CM Security roles

FMT_SMR.1.1/CM The TSF shall maintain the roles **[Card Manager]**.

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 *Refinement:* 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 and installing a new application package on the card]**.
Application note:

- There is no dynamic package loading on the Java Card platform. New packages can be loaded and installed on the card only on demand of the card issuer.

8.1.6 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. The requirements are expressed in terms of security functional requirements from [CC-2].

FPT_TST.1/SCP TSF Testing

FPT_TST.1.1/SCP The TSF shall run a suite of self tests [selection: periodically during normal operation] to demonstrate the correct operation of **[security mechanisms of the IC].**

FPT_TST.1.2/SCP The TSF shall provide authorized users with the capability to verify the integrity of **[Keys]**.

FPT_TST.1.3/SCP The TSF shall provide authorized users with the capability to verify the integrity of **[Applets, user PIN, user Keys]**.

FPT_PHP.3/SCP Resistance to physical attacks

FPT_PHP.3.1/SCP The TSF shall resist **[physical attacks]** to the **[TOE]** by responding automatically such that the TSP is not violated.

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.

8.1.7 CMGR Group Security Functional Requirements

This group includes requirements for Card Manager.
The requirements are expressed in terms of security functional requirements from [CC-2].

FDP_ACC.1/CMGR Subset access control

FDP_ACC.1.1/CMGR The TSF shall enforce the **[CARD CONTENT MANAGEMENT access control SFP]** on **[loading of code and keys by the Operator]**.

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:
[Subjects: Byte Code Verifier, Operator, Issuer, Card Manager
Objects: applets and keys
Security Attributes: DAP for applets; type and KEK for keys.]

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:
[The Card Manager loads applets into the card on behalf of the Byte Code Verifier.
The Card Manager extradites applets in the card on behalf of the Operator.
The Card Manager locks the loading of applets on the card on behalf of the Issuer.
The Card Manager loads GP keys into the cards on behalf of the Operator.]
.
FDP_ACF.1.3/CMGR The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **[none]**.

FDP_ACF.1.4/CMGR The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[No code but Java packages can be loaded or deleted]**.

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 [selection: modify] the security attributes **[code category]** to **[none]**.

FMT_MSA.3/CMGR Static attribute initialization

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

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

8.1.8 ASFR Group Security Functional Requirements

This group includes specific requirements for the TOE.
The requirements are expressed in terms of security functional requirements from [CC-2].

FPT_FLS.1/SpecificAPI Failure with preservation of secure state

FPT_FLS.1.1/SpecificAPI The TSF shall preserve a secure state when the following types of failures occur: **[the application fails to perform a specific execution flow control protected by the Specific API]**.

FPT_ITT.1/SpecificAPI Basic internal TSF data transfer protection

FPT_ITT.1.1/SpecificAPI The TSF shall protect TSF data from [selection: disclosure and modification] when it is transmitted between separate parts of the TOE.

FPR_UNO.1/SpecificAPI Unobservability

FPR_UNO.1.1/SpecificAPI The TSF shall ensure that **[external attacker]** are unable to observe the operation **[as sensitive comparison or copy]** on **[sensitive objects defined]** by **[the application using the Specific API]**.

FCS_RND.1 Quality metric for random numbers

FCS_RND.1.1 The TSF shall provide a mechanism to generate random numbers that meet **[RGS-B1].**

8.2 SECURITY ASSURANCE REQUIREMENTS

The security assurance requirements are based on the package level EAL5 augmented with AVA_VAN.5 and ALC_DVS.2, according to [CC-3].

8.3 SECURITY REQUIREMENTS RATIONALE

8.3.1 OBJECTIVES

	O.SID	O.OPERATE	O.RESOURCES	O.FIREWALL	O.NATIVE	O.REALLOCATION	O.GLOBAL_ARRAYS_CONFID	O.GLOBAL_ARRAYS_INTEG	O.ALARM	O.TRANSACTION	O.CIPHER	O.PIN_MNGT	O.KEY_MNGT	O.OBJ_DELETION	O.INSTALL	O.LOAD	O.DELETION	O.SCP.RECOVERY	O.SCP.SUPPORT	O.SCP.IC	O.CARD_MANAGEMENT	O.SpecificAPI	O.RND
FDP_IFC.1/JCVM				x			x	x															
FDP_IFF.1/JCVM				x			x	x															
FDP_RIP.1/OBJECTS						x				x		x	x										
FMT_MSA.2/FIREWALL_JCVM				x																			
FMT_MSA.3/FIREWALL	x			x																			
FMT_MSA.3/JCVM	x			x																			
FMT_SMR.1/JCRE			x	x																			
FMT_SMF.1/CORE_LC			x	x																			
FCS_CKM.1/RSA Std											x		x										
FCS_CKM.1/RSA CRT											x		x										
FCS_CKM.1/GP											x		x										
FCS_CKM.1/DH											x		x										
FCS_CKM.2/RSA											x		x										
FCS_CKM.2/TDES											x		x										
FCS_CKM.2/AES											x		x										
FCS_CKM.2/DH											x		x										
FCS_CKM.2/STORE Data											x		x										
FCS_CKM.2/PUT KEY											x		x										
FCS_CKM.3/RSA											x		x										
FCS_CKM.3/TDES											x		x										
FCS_CKM.3/AES											x		x										
FCS_CKM.3/DH											x		x										
FCS_CKM.4											x		x										
FCS_COP.1/RSA-SIGN											x		x										
FCS_COP.1/RSA-CIPHER											x		x										
FCS_COP.1/TDES-CIPHER											x		x										
FCS_COP.1/AES-CIPHER											x		x										
FCS_COP.1/TDES-MAC											x		x										
FCS_COP.1/AES-CMAC											x		x										
FCS_COP.1/SHA											x		x										
FDP_RIP.1/APDU						x	x			x		x	x										

	O.SID	O.OPERATE	O.RESOURCES	O.FIREWALL	O.NATIVE	O.REALLOCATION	O.GLOBAL_ARRAYS_CONFID	O.GLOBAL_ARRAYS_INTEG	O.ALARM	O.TRANSACTION	O.CIPHER	O.PIN_MNGT	O.KEY_MNGT	O.OBJ_DELETION	O.INSTALL	O.LOAD	O.DELETION	O.SCP.RECOVERY	O.SCP.SUPPORT	O.SCP.IC	O.CARD_MANAGEMENT	O.SpecificAPI	O.RND
FDP_RIP.1/bArray						X	X			X		X	X										
FDP_RIP.1/ABORT						X				X		X	X										
FDP_RIP.1/KEYS						X				X		X	X										
FDP_ROL.1/FIREWALL		X	X							X		X											
FAU_ARP.1		X	X						X														
FDP_SDI.2												X	X										
FPT_TDC.1		X																					
FPT_FLS.1/JCS		X	X						X														
FPR_UNO.1											X	X	X										
FMT_MTD.1/JCRE	X		X	X																			
FMT_MTD.3/JCRE	X		X	X																			
FIA_ATD.1/AID	X	X																					
FIA_UID.2/AID	X																						
FIA_USB.1/AID	X	X																					
FDP_ITC.2/Installer	X	X		X											X								
FMT_SMR.1/Installer			X	X																			
FPT_FLS.1/Installer		X	X						X						X								
FPT_RCV.3/Installer		X	X												X		X						
FMT_MSA.1/ADEL	X			X													X						
FMT_MSA.3/ADEL	X			X													X						
FMT_SMR.1/ADEL			X	X													X						
FMT_SMF.1/ADEL	X		X	X																			
FDP_ACC.2/ADEL																	X						
FDP_ACF.1/ADEL																	X						
FDP_RIP.1/ADEL						X				X		X	X				X						
FPT_FLS.1/ADEL		X	X						X								X						
FDP_ACC.2/FIREWALL		X		X								X											
FDP_ACF.1/FIREWALL		X		X	X							X											
FMT_MSA.1/JCRE	X			X																			
FMT_MSA.1/JCVM	X			X																			
FDP_RIP.1/TRANSIENT						X				X		X	X										
FDP_RIP.1/ODEL						X				X		X	X	X									
FPT_FLS.1/ODEL		X	X						X					X									
FMT_MSA.1/CM	X			X																			
FMT_MSA.3/CM	X			X																			
FMT_SMR.1/CM			X	X																			
FMT_SMF.1/CM	X		X	X																	X		
FCO_NRO.2/CM																X							
FIA_UAU.1/CM																X							
FIA_UID.1/CM																X							
FDP_IFC.2/CM																X							

	O.SID	O.OPERATE	O.RESOURCES	O.FIREWALL	O.NATIVE	O.REALLOCATION	O.GLOBAL_ARRAYS_CONFID	O.GLOBAL_ARRAYS_INTEG	O.ALARM	O.TRANSACTION	O.CIPHER	O.PIN_MNGT	O.KEY_MNGT	O.OBJ_DELETION	O.INSTALL	O.LOAD	O.DELETION	O.SCP.RECOVERY	O.SCP.SUPPORT	O.SCP.IC	O.CARD_MANAGEMENT	O.SpecificAPI	O.RND
FDP_IFF.1/CM																X							
FDP_UIT.1/CM																X							
FTP_ITC.1/CM																X							
FPT_TST.1/SCP																			X				
FPT_PHP.3/SCP																				X			
FPT_RCV.4/SCP																		X					
FDP_ACC.1/CMGR																					X		
FDP_ACF.1/CMGR																					X		
FMT_MSA.1/CMGR																					X		
FMT_MSA.3/CMGR																					X		
FPT_FLS.1/SpecificAPI																						X	
FPT_ITT.1/SpecificAPI																						X	
FPR_UNO.1/SpecificAPI																						X	
FCS_RND.1																							X

Table 9: rationale objective vs. SFR

8.3.1.1 SECURITY OBJECTIVES FOR THE TOE

8.3.1.1.1 IDENTIFICATION

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

8.3.1.1.2 EXECUTION

O.OPERATE The TOE is protected in various ways against applets' actions (security architecture described in ADV_ARC.1, FPT_TDC.1), the FIREWALL access control policy (FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL), and is able to detect and block various failures or security violations during usual working (FPT_FLS.1/ADEL, FPT_FLS.1/JCS, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FAU_ARP.1). 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.RESOURCES The TSFs detects stack/memory overflows during execution of applications (FAU_ARP.1, FPT_FLS.1/ADEL, FPT_FLS.1/JCS, 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/JCRE, FMT_SMF.1/CORE_LC FMT_SMR.1/ADEL, FMT_SMF.1/ADEL, FMT_SMF.1/CM, and FMT_SMR.1/CM).

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

O.NATIVE The JCVM is the machine running the bytecode of the applets. These can only be linked with API methods or other packages already on the card. This objective mainly relies on the objective OE.APPLET, which upholds the assumption A.APPLET. The description of how the security architecture of the TSF (ADV_ARC.1) covers this objective. This security objective is also satisfied by the FDP_ACF.1/FIREWALL SFR.

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, and FDP_RIP.1/ADEL, which imposes that the contents of the re-allocated block shall always be cleared before delivering the block.

O.GLOBAL_ARRAYS_CONFID Only arrays can be designated as global, and the only global arrays required in the Java Card API are the APDU buffer and the byte array input parameter (bArray) to an applet's install method. The clearing requirement of 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.

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

8.3.1.1.3 SERVICES

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

O.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).
O.CIPHER This security objective is directly covered by FCS_CKM.1/RSA Std, FCS_CKM.1/RSA CRT, FCS_CKM.1/GP, FCS_CKM.1/DH, FCS_CKM.2/RSA, FCS_CKM.2/TDES, FCS_CKM.2/AES, FCS_CKM.2/DH, FCS_CKM.2/STORE Data, FCS_CKM.2/PUT KEY, FCS_CKM.3/RSA, FCS_CKM.3/TDES, FCS_CKM.3/AES, FCS_CKM.3/DH, FCS_CKM.4 and FCS_COP.1/RSA-SIGN, FCS_COP.1/RSA-CIPHER, FCS_COP.1/TDES-CIPHER, FCS_COP.1/AES-CIPHER, FCS_COP.1/TDES-MAC, FCS_COP.1/AES-CMAC, FCS_COP.1/SHA. The SFR FPR_UNO.1 contributes in covering this security objective and controls the observation of the cryptographic operations which may be used to disclose the keys.

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

O.KEY-MNGT This relies on the same security functional requirements as O.CIPHER, plus FDP_RIP.1 and FDP_SDI.2 as well. Precisely it is met by the following components: FCS_CKM.1/RSA Std, FCS_CKM.1/RSA CRT, FCS_CKM.1/GP, FCS_CKM.1/DH, FCS_CKM.2/RSA, FCS_CKM.2/TDES, FCS_CKM.2/AES, FCS_CKM.2/DH, FCS_CKM.2/STORE Data, FCS_CKM.2/PUT KEY, FCS_CKM.3/RSA, FCS_CKM.3/TDES, FCS_CKM.3/AES, FCS_CKM.3/DH, FCS_CKM.4, FCS_COP.1/RSA-SIGN, FCS_COP.1/RSA-CIPHER,

FCS_COP.1/TDES-CIPHER, FCS_COP.1/AES-CIPHER, FCS_COP.1/TDES-MAC, FCS_COP.1/AES-CMAC, FCS_COP.1/SHA, FPR_UNO.1, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL and FDP_RIP.1/TRANSIENT.

8.3.1.1.4 OBJECT DELETION

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

8.3.1.1.5 APPLET MANAGEMENT

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

O.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 and authentication (FIA_UAU.1/CM, FIA_UID.1/CM) and transmission mechanisms are also enforced (FTP_ITC.1/CM).

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, and FMT_SMR.1/ADEL) included in the group ADELG also contribute to meet this objective.

8.3.1.1.6 SCP

O.SCP.RECOVERY This security objective specifies that the platform must behave securely if an unexpected loss of power occurs. This is covered by FPT_RCV.4/SCP which specifies the recovery after unexpected power failure.
O.SCP.SUPPORT This security objective specifies that the SCP provides security features to the JCS. This is provided by FPT_TST.1/SCP. This is also provided by requirements of the IC, which are described in [ST-IC].
O.SCP.IC This security objective specifies that the IC must provide mechanisms to protect itself against physical attacks. This is provided by FPT_PHP.3/SCP. This is also provided by requirements of the IC, which are described in [ST-IC].

8.3.1.1.7 Card Management

O.CARD_MANAGEMENT This security objective specifies that the access control to card management functions. This is enforced by FDP_ACC.1/CMGR, FDP_ACF.1/CMGR, FMT_MSA.1/CMGR, FMT_MSA.3/CMGR, FMT_SMF.1/CM.

8.3.1.1.8 ASFR

O. SpecificAPI The security objective is met by the following SFR FPT_FLS.1/SpecificAPI, FPT_ITT.1/SpecificAPI and FPR_UNO.1/SpecificAPI.

O.RND The security objective O.RND is met by the following SFR FCS_RND.1.

8.3.2 DEPENDENCIES

8.3.2.1 SFRS DEPENDENCIES

Requirements	CC dependencies	Satisfied dependencies
FAU_ARP.1	FAU_SAA.1	Unsupported
FCO_NRO.2/CM	FIA_UID.1	FIA_UID.1/CM
FCS_CKM.1/RSA Std	(FCS_CKM.2 or FCS_COP.1), FCS_CKM.4	FCS_CKM.2/RSA, FCS_CKM.4
FCS_CKM.1/RSA CRT	(FCS_CKM.2 or FCS_COP.1), FCS_CKM.4	FCS_CKM.2/RSA, FCS_CKM.4
FCS_CKM.1/GP	(FCS_CKM.2 or FCS_COP.1), FCS_CKM.4	FCS_CKM.2/PUT KEY FCS_CKM.2/STORE DATA FCS_CKM.4
FCS_CKM.1/DH	(FCS_CKM.2 or FCS_COP.1), FCS_CKM.4	FCS_CKM.2/DH, FCS_CKM.4
FCS_CKM.2/RSA	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/RSA Std FCS_CKM.1/RSA CRT FCS_CKM.4
FCS_CKM.2/TDES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/GP FCS_CKM.4
FCS_CKM.2/AES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/GP FCS_CKM.4
FCS_CKM.2/DH	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/DH FCS_CKM.4
FCS_CKM.2/STORE Data	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/GP FCS_CKM.4
FCS_CKM.2/PUT KEY	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/GP CS_CKM.4
FCS_CKM.3/RSA	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/RSA Std FCS_CKM.1/RSA CRT FCS_CKM.4
FCS_CKM.3/TDES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/GP, FCS_CKM.4
FCS_CKM.3/AES	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/GP, FCS_CKM.4
FCS_CKM.3/DH	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/DH FCS_CKM.4
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2	FCS_CKM.1/RSA Std FCS_CKM.1/RSA CRT FCS_CKM.1/GP FCS_CKM.1/DH
FCS_COP.1/RSA-SIGN	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/RSA Std FCS_CKM.1/RSA CRT FCS_CKM.4
FCS_COP.1/RSA-CIPHER	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/RSA Std FCS_CKM.1/RSA CRT FCS_CKM.4
FCS_COP.1/TDES-CIPHER	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/GP, FCS_CKM.4
FCS_COP.1/AES-CIPHER	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/GP, FCS_CKM.4

Requirements	CC dependencies	Satisfied dependencies
FCS_COP.1/TDES-MAC	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/GP, FCS_CKM.4
FCS_COP.1/AES-CMAC	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1/GP, FCS_CKM.4
FCS_COP.1/SHA	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FDP_ITC.2/Installer, FCS_CKM.4
FDP_ACC.2/ADEL	FDP_ACF.1	FDP_ACF.1/ADEL
FDP_ACC.2/FIREWALL	FDP_ACF.1	FDP_ACF.1/FIREWALL
FDP_ACF.1/ADEL	FDP_ACC.1, FMT_MSA.3	FDP_ACC.2/ADEL , FMT_MSA.3/ADEL
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_IFC.2/CM	FDP_IFF.1	FDP_IFF.1/CM
FDP_IFF.1/JCVM	FDP_IFC.1, FMT_MSA.3	FDP_IFC.1/JCVM, FMT_MSA.3/JCVM
FDP_IFF.1/CM	FDP_IFC.1, FMT_MSA.3	FDP_IFC.2/CM, FMT_MSA.3/CM
FDP_ITC.2/Installer	(FDP_ACC.1 or FDP_IFC.1), FPT_TDC.1, (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM, FPT_TDC.1
FDP_RIP.1/OBJECTS	none	
FDP_RIP.1/APDU	none	
FDP_RIP.1/bArray	none	
FDP_RIP.1/ABORT	none	
FDP_RIP.1/KEYS	none	
FDP_RIP.1/ADEL	none	
FDP_RIP.1/TRANSIENT	none	
FDP_RIP.1/ODEL	none	
FDP_ROL.1/FIREWALL	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM
FDP_SDI.2	none	
FIA_ATD.1/AID	none	
FIA_UAU.1/CM	FIA_UID.1	FIA_UID.1/CM
FIA_UID.1/CM	none	
FIA_UID.2/AID	none	
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_USB.1/AID	FIA_ATD.1	FIA_ATD.1/AID
FMT_MSA.1/ADEL	(FDP_ACC.1 or FDP_IFC.1), FMT_SMF.1, FMT_SMR.1	FDP_ACC.2/ADEL, FMT_SMF.1/ADEL, FMT_SMR.1/ADEL
FMT_MSA.1/JCVM	(FDP_ACC.1 or FDP_IFC.1), FMT_SMF.1, FMT_SMR.1	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_SMF.1/CORE_LC, FMT_SMR.1/JCRE
FMT_MSA.1/JCRE	(FDP_ACC.1 or FDP_IFC.1), FMT_SMF.1, FMT_SMR.1	FDP_IFC.1/JCVM, FMT_SMR.1/JCRE, FMT_SMF.1/CORE_LC, FDP_ACC.2/FIREWALL
FMT_MSA.1/CM	(FDP_ACC.1 or FDP_IFC.1), FMT_SMF.1, FMT_SMR.1	FDP_IFC.2/CM, FMT_SMR.1/CM, FMT_SMF.1/CM

Requirements	CC dependencies	Satisfied dependencies
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/JCRE
FMT_MSA.3/FIREWALL	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1/JCRE
FMT_MSA.3/JCVM	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1/JCVM, FMT_SMR.1/JCRE
FMT_MSA.3/ADEL	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1/ADEL, FMT_SMR.1/ADEL
FMT_MSA.3/CM	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1/CM, FMT_SMR.1/CM
FMT_MTD.1/JCRE	FMT_SMF.1, FMT_SMR.1	FMT_SMR.1/JCRE, FMT_SMF.1/CORE_LC
FMT_MTD.3/JCRE	FMT_MTD.1	FMT_MTD.1/JCRE
FMT_SMR.1/JCRE	FIA_UID.1	FIA_UID.2/AID
FMT_SMR.1/Installer	FIA_UID.1	Unsupported
FMT_SMR.1/ADEL	FIA_UID.1	Unsupported
FMT_SMR.1/CM	FIA_UID.1	FIA_UID.1/CM
FMT_SMF.1/CORE_LC	none	
FMT_SMF.1/ADEL	none	
FMT_SMF.1/CM	none	
FPR_UNO.1	none	
FPT_FLS.1/JCS	none	
FPT_FLS.1/Installer	none	
FPT_FLS.1/ADEL	none	
FPT_FLS.1/ODEL	none	
FPT_RCV.3/Installer	AGD_OPE.1	AGD_OPE.1
FPT_TDC.1	none	
FTP_ITC.1/CM	none	
FPT_TST.1/SCP	none	
FPT_PHP.3/SCP	none	
FPT_RCV.4/SCP	none	
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_SMF.1, FMT_SMR.1	FDP_ACC.1/CMGR, FMT_SMF.1/CM, FMT_SMR.1/CM
FMT_MSA.3/CMGR	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1/CMGR, FMT_SMR.1/CM
SFR FPT_FLS.1/SpecificAPI	none	
FPT_ITT.1/SpecificAPI	none	
FPR_UNO.1/SpecificAPI.	none	
FCS_RND.1	none	

8.3.2.1.1 RATIONALE FOR THE EXCLUSION OF DEPENDENCIES

The dependency FIA_UID.1 of FMT_SMR.1/Installer is unsupported. This is required by the component FMT_SMR.1 in group InstG. However, the role installer defined in this component is attached to an IT security function rather than to a "user" of the CC terminology. The installer does not "identify" itself with respect to the TOE, but is a part of it. Thus, here it is claimed that this dependency can be left out. The reader may notice

that the role is required because of the SFRs on management of TSF data and security attributes, essentially those of the firewall policy.

The dependency FAU_SAA.1 of FAU_ARP.1 is unsupported. Potential violation analysis is used to specify the set of auditable events whose occurrence or accumulated occurrence held to indicate a potential violation of the SFRs, and any rules to be used to perform the violation analysis. The dependency of FAU_ARP.1 on this functional requirement assumes that a "potential security violation" is an audit event indicated by the FAU_SAA.1 component. The events listed in FAU_ARP.1 are, on the contrary, merely self-contained ones (arithmetic exception, ill-formed bytecodes, access failure) and ask for a straightforward reaction of the TSFs on their occurrence at runtime. The JCVM or other components of the TOE detect these events during their usual working order. Thus, in principle there would be no applicable audit recording in this framework. Moreover, no specification of one such recording is provided elsewhere. Therefore no set of auditable events could possibly be defined.

The dependency FIA_UID.1 of FMT_SMR.1/ADEL is unsupported. This is required by the component FMT_SMR.1 in group ADELG. However, the role applet deletion manager defined in this component is attached to an IT security function rather than to a "user" of the CC terminology. The installer does not "identify" itself with respect to the TOE, but is a part of it. Thus, here it is claimed that this dependency can be left out. The reader may notice that the role is required because of the SFRs on management of TSF data and security attributes, essentially those of the firewall policy.

8.3.2.2 SARS DEPENDENCIES

Requirements	CC dependencies	Satisfied dependencies
ASE_CCL.1	ASE_INT.1, ASE_ECD.1, ASE_REQ.1	ASE_INT.1, ASE_ECD.1, ASE_REQ.2
ASE_ECD.1	None	
ASE_INT.1	None	
ASE_OBJ.2	ASE_SPD.1	ASE_SPD.1
ASE_REQ.2	ASE_OBJ.2, ASE_ECD.1	ASE_OBJ.2, ASE_ECD.1
ASE_SPD.1	None	
ASE_TSS.1	ASE_INT.1, ASE_REQ.1, ADV_FSP.1	ASE_INT.1, ASE_REQ.2, ADV_FSP.5
ADV_ARC.1	ADV_FSP.1; ADV_TDS.1	ADV_FSP.5; ADV_TDS.4
ADV_FSP.5	ADV_TDS.1; ADV_IMP.1	ADV_TDS.4; ADV_IMP.1
ADV_IMP.1	ADV_TDS.3; ALC_TAT.1	ADV_TDS.4; ALC_TAT.2
ADV_INT.2	ADV_IMP.1; ADV_TDS.3; ALC_TAT.1	ADV_IMP.1; ADV_TDS.4; ALC_TAT.2
ADV_TDS.4	ADV_FSP.5	ADV_FSP.5
AGD_OPE.1	ADV_FSP.1	ADV_FSP.5
AGD_PRE.1	None	
ALC_CMC.4	ALC_CMS.1; ALC_DVS.1; ALC_LCD.1	ALC_CMS.5; ALC_DVS.2; ALC_LCD.1
ALC_CMS.5	None	
ALC_DEL.1	None	
ALC_DVS.2	None	
ALC_LCD.1	None	
ALC_TAT.2	ADV_IMP.1	ADV_IMP.1
ATE_COV.2	ADV_FSP.2; ATE_FUN.1	ADV_FSP.5; ATE_FUN.1
ATE_DPT.3	ADV_ARC.1; ADV_TDS.4; ATE_FUN.1	ADV_ARC.1; ADV_TDS.4; ATE_FUN.1
ATE_FUN.1	ATE_COV.1	ATE_COV.2
ATE_IND.2	ADV_FSP.2; AGD_OPE.1; AGD_PRE.1; ATE_COV.1; ATE_FUN.1	ADV_FSP.5; AGD_OPE.1; AGD_PRE.1; ATE_COV.2; ATE_FUN.1
AVA_VAN.5	ADV_ARC.1; ADV_FSP.4; ADV_TDS.3; ADV_IMP.1; AGD_OPE.1; AGD_PRE.1; ATE_DPT.1	ADV_ARC.1; ADV_FSP.5; ADV_TDS.4; ADV_IMP.1; AGD_OPE.1; AGD_PRE.1; ATE_DPT.3

8.3.3 RATIONALE FOR THE SECURITY ASSURANCE REQUIREMENTS

8.3.3.1 EAL5: Semi-formally designed and tested

EAL5 is required for this type of TOE and product since it is intended to defend against sophisticated attacks. This evaluation assurance level allows a developer to gain maximum assurance from positive security engineering based on good practices. 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.Additionally the semi-formal methodology, provided by EAL5, gives more confidence in the design of the TOE

8.3.3.2 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.4, ADV_TDS.3, ADV_IMP.1, AGD_OPE.1, AGD_PRE.1, and ATE_DPT.1. All of them are satisfied by EAL5.

8.3.3.3 *ALC DVS.2 SECURITY MEASURES*

Development security is concerned with physical, procedural, personnel and other technical measures that may be used in the development environment to protect the TOE and the embedding product. The standard ALC_DVS.1 requirement mandated by EAL5 is not enough. Due to the nature of the TOE and embedding product, it is necessary to justify the sufficiency of these procedures to protect their confidentiality and integrity. ALC_DVS.2 has no dependencies.

8.3.4 Compatibility between SFR of [ST-JCS] and [ST-IC]

The following table lists the SFRs that are declared on the Centaurus FB 04 Integrated Circuit Security Target [ST-IC] and separates them in:

> **IP_SFR**: Irrelevant Platform-SFRs not being used by the Composite-ST.
> **RP_SFR-SERV**: Relevant Platform-SFRs being used by the Composite-ST to implement a security service with associated TSFI.
> **MRP_SFR-MECH**: Relevant Platform-SFRs being used by the Composite-ST because of its security properties providing protection against attacks to the TOE as a whole and are addressed in ADV_ARC. These required security properties are a result of the security mechanisms and services that are implemented in the Platform TOE, as specified in [JIL_CPE].

These definitions are according to the [JIL_CPE] on which the Platform TOE on our case is the relaying IC, the Centaurus FB 04 Integrated Circuit.

The first column lists the Centaurus FB 04 SFRs and the next columns indicate their classification according to the paragraph above. The SFR's on the cells of the classification belong the MultiApp Essential v1.1 TOE described in this document. If there is no SFR on each cell is because not all CC class families have a corresponding match on both sides, but all SFRs from the Centaurus FB 04 have been classified. Moreover, no contradictions have been found between the Platform-SFRs set and the SFRs related to the composite product.

CENTAURUS_FB_04 SFR's	IP_SFR (I)	RP_SFR-SERV (S)	RP_SFR-MECH (M)
FRU_FLT.2	X		
FPT_FLS.1			X FPT_FLS.1/JCS FPT_FLS.1/Installer FPT_FLS.1/ADEL FPT_FLS.1/ODEL FPT_FLS.1/SpecificAPI
FMT_LIM.1			X FMT_MTD.3/JCRE
FMT_LIM.2			X FMT_MTD.1/JCRE
FAU_SAS.1	X		
FDP_SDC.1	X		
FDP_SDI.2/RAM	X		
FDP_SDI.2/NVM			X FDP_SDI.2

CENTAURUS_FB_04 SFR's	IP_SFR (I)	RP_SFR-SERV (S)	RP_SFR-MECH (M)
FDP_SDI.2/Register&Bus	X		
FPT_PHP.3			X FPT_PHP.3/SCP
FDP_ITT.1	X		
FPT_ITT.1	X		
FDP_IFC.1			X FDP_IFC.1/JCVM
FCS_RNG.1/RGS-IC		X FCS_RND.1	
FCS_RNG.1/PTG.2	X		
FIA_API.1			X FIA_UID.2/AID FIA_UAU.1/CM FIA_UID.1/CM
FMT_LIM.1/Loader	X		
FMT_LIM.2/Loader	X		
FDP_UIT.1			X FDP_UIT.1/CM
FDP_ACC.1/Loader	X		
FDP_ACF.1/Loader	X		
FTP_TRP.1			X FTP_ITC.1/CM
FDP_ACC.1		X FDP_ACC.1/CMGR	
FDP_ACF.1		X FDP_ACF.1/CMGR	
FMT_MSA.1		X FMT_MSA.1/ADEL FMT_MSA.1/JCRE FMT_MSA.1/JCVM FMT_MSA.1/CM FMT_MSA.1/CMGR	
FMT_MSA.3		X FMT_MSA.3/FIREWALL FMT_MSA.3/JCVM FMT_MSA.3/ADEL FMT_MSA.3/CM FMT_MSA.3/CMGR	
FMT_SMF.1		X FMT_SMF.1/CORE_LC FMT_SMF.1/ADEL FMT_SMF.1/CM	

Table 10: Compatibility [ST-IC] and [ST-JCS] SFRs

9 TOE SUMMARY SPECIFICATION

9.1 TOE SECURITY FUNCTIONS

TOE Security Functions are provided by the TOE embedded software (including the optional NVM ES) and by the chip.

9.1.1 SF provided by MultiApp Essential v1.1 platform

9.1.1.1 SF.FW: Firewall

The JCRE firewall enforces applet isolation. The *JCRE* shall allocate and manage a context for each *applet* or *package* installed respectively loaded on the card and its own JCRE context. *Applet* cannot access each other's objects unless they are defined in the same package (they share the same context) or they use the object sharing mechanism supported by *JCRE*.

An operation OP.PUT (S1, S.MEMBER, I.DATA) is allowed if and only if the active context is "JCRE"; other OP.PUT operations are allowed regardless of the active context's value.	FDP_IFC.1/JCVM FDP_IFF.1/JCVM
Upon allocation of a resource to class instances and arrays, any previous information content of the resource is made unavailable	FDP_RIP.1/OBJECTS
Only the S.JCRE can modify the security attributes the active context, the selected applet context security attributes.	FMT_MSA.1/JCRE
Only the S.JCVM can modify the security attributes the active context, the currently active Context and the Active Applets security attributes.	FMT_MSA.1/JCVM FMT_MSA.3/JCVM
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.	FMT_MSA.2/FIREWALL_JCVM
provide restrictive default values for security attributes that are used to enforce the SFP.	FMT_MSA.3/FIREWALL
The TSF shall maintain the roles: the Java Card RE, the Java Card VM. The TSF shall be able to associate users with roles.	FMT_SMR.1/JCRE
The TSF shall be capable of performing the following management functions: • Modify the active context and the SELECTed applet Context. • Modify the list of registered applets' AID	FMT_SMF.1/CORE_LC
([JCRE304]§6.2.8) An S.PACKAGE may freely perform any of OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW or OP.TYPE_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
([JCRE304]§6.2.8) An S.PACKAGE may freely perform any of OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE or OP.THROW upon any 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.	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
([JCRE304]§6.2.8.10) An S.PACKAGE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT whose Sharing attribute has value "SIO" only if O.JAVAOBJECT	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL

is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.	
• ([JCRE304], §6.2.8.6,) An S.PACKAGE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if one of the following applies: (c) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable", (d) The value of the attribute Selection Status of the package whose AID is "Package AID' is "Non-multiselectable", and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute ActiveApplets, and in either of the cases above the invoked interface method extends the Shareable interface	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
An S.PACKAGE may perform an OP.CREATE only if the value of the Sharing parameter(*) is "Standard".	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
The subject S.JCRE can freely perform OP.JAVA(...) and OP.CREATE, with the following two exceptions: 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 with OP.CREATE and a "CLEAR_ON_DESELECT" LifeTime parameter if the active context is not the same as the SELECTed applet Context.	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
Upon deallocation of the resource from any transient object, any previous information content of the resource is made unavailable.	FDP_RIP.1/TRANSIENT
The TSF shall permit the rollback of the operations OP.JAVA and OP.CREATE on the O.JAVAOBJECTs.	FDP_ROL.1/FIREWALL
The TSF shall permit operations to be rolled back within the scope of a select(), deselect(), process() or install() call, notwithstanding the restrictions given in [JCRE304], §7.7, within the bounds of the Commit Capacity ([JCRE304], §7.8), and those described in [JCAPI304].	FDP_ROL.1/FIREWALL
Only updates to persistent objects participate in the transaction. Updates to transient objects and global arrays are never undone, regardless of whether or not they were “inside a transaction.” [JCRE304], §7.7	FDP_ROL.1/FIREWALL
A TransactionException is thrown if the commit capacity is exceeded during a transaction. [JCRE304], §7.8	FDP_ROL.1/FIREWALL
Transaction & PIN: When comparing a PIN, even if a transaction is in progress, update of internal state - the try counter, the validated flag, and the blocking state, do not participate in the transaction. [JCAPI304]	FDP_ROL.1/FIREWALL

9.1.1.2 SF.API: Application Programming Interface

This security function provides the cryptographic algorithm and functions used by the TSF:

- TDES algorithm support 112-bit key and 168-bit key
- RSA algorithm supports up to 2048 bit keys (Std method) or up to 3072 (CRT method).
- AES algorithm with 128, 192 and 256 bit keys.
- Random generator uses the certified Hardware Random Generator that fulfils the requirements of AIS31 (see [ST-IC]).
- SHA224, SHA-256, SHA-384, SHA-512 algorithms
- Diffie-Hellman based on exponentiation.

This security function controls all the operations relative to the card keys management.

- Key generation: The TOE provides the following:
  - RSA key generation manages 2048-bits long keys (and up to 3072 for CRT only). The RSA key generation is SW and does not use the IC cryptographic library.
  - The TDES key generation (for session keys) uses the random generator.
  - AES key generation
  - DH key generation
- Key destruction: the TOE provides a specified cryptographic key destruction method that makes Key unavailable.

This security function ensures the confidentiality of keys during manipulation and ensures the de-allocation of memory after use.
This security function is supported by the IC security function SF_CRYPTO (Cryptographic Services) and for SF_RNG (Random Number Generator) (see [ST-IC]).

RSA standard Key generation Algorithm - 2048	FCS_CKM.1/RSA Std
RSA CRT Key generation Algorithm - 2048, 3072	FCS_CKM.1/RSA CRT
TDES Key generation Algorithm - 112, 168	FCS_CKM.1/GP
AES Key generation Algorithm - 128, 192, 256	FCS_CKM.1/GP
DH Key agreement Algorithm 2048	FCS_CKM.1/DH
Key distribution with JC API specification for applications	FCS_CKM.2/RSA FCS_CKM.2/TDES FCS_CKM.2/AES FCS_CKM.2/DH
Key distribution with Gemalto API specification for Global Platform	FCS_CKM.2/STORE Data FCS_CKM.2/PUT KEY
Key access with JC API getkey()	FCS_CKM.3/RSA FCS_CKM.3/TDES FCS_CKM.3/AES FCS_CKM.3/DH
Key deletion with JC API clearkey()	FCS_CKM.4
RSA standard Signature & Verification – RSA SHA PKCS#1, RSA SHA PKCS#1 PSS –2048	FCS_COP.1/RSA-SIGN
RSA CRT Signature & Verification – RSA SHA PKCS#1, RSA SHA PKCS#1 PSS 2048, 3072	FCS_COP.1/RSA-SIGN
RSA standard Encryption & Decryption –2048	FCS_COP.1/RSA-CIPHER
RSA CRT Encryption & Decryption –2048, 3072	FCS_COP.1/RSA-CIPHER
TDES Encryption & Decryption – DES NOPAD, DES PKCS#5, DES 9797 M1 M2 – 112, 168	FCS_COP.1/TDES-CIPHER
TDES Signature & Verification – DES MAC ISO9797-1 M1 M2, M3 DES MAC NOPAD, DES MAC PKCS#5- 112, 168	FCS_COP.1/TDES-MAC
AES Encryption & Decryption – AES 128 NOPAD – 128, 192, 256	FCS_COP.1/AES-CIPHER
AES Signature & Verification – AES 128 NOPAD – 128, 192, 256	FCS_COP.1/AES-CMAC
SHA-224, SHA-256, SHA-384, SHA-512 Message digest	FCS_COP.1/SHA

9.1.1.3 SF.CSM: Card Security Management

Upon allocation of a resource to the APDU buffer, any previous information content of the resource is made unavailable.	FDP_RIP.1/APDU

Upon deallocation of a resource from the bArray object, any previous information content of the resource is made unavailable.	FDP_RIP.1/bArray
Upon deallocation of a resource from any reference to an object instance created during an aborted transaction, any previous information content of the resource is made unavailable.	FDP_RIP.1/ABORT
Upon deallocation of a resource from the cryptographic buffer (D.CRYPTO), any previous information content of the resource is made unavailable.	FDP_RIP.1/KEYS
The TSF shall take the following actions: • throw an exception, • or lock the card session • or reinitialize the Java Card System and its data upon detection of a potential security violation.	FAU_ARP.1
The TOE detects the following potential security violation: • CAP file inconsistency • Applet life cycle inconsistency • Card Manager life cycle inconsistency • Card tearing (unexpected removal of the Card out of the CAD) and power failure • Abortion of a transaction in an unexpected context (see abortTransaction(), [JCAPI304] and ([JCRE304], §7.6.2) • Violation of the Firewall or JCVM SFPs • Unavailability of resources • Array overflow • Random trap detection	FAU_ARP.1
The TSF shall monitor user data stored in containers controlled by the TSF for **integrity errors** on all the following objects: Cryptographic keys, PINs, applets, and softmasks when they are stored in EEPROM. Upon detection of a data integrity error, the TSF shall: • Prevent the use of modified data • Raise an exception	FDP_SDI.2
In order to consistently interpret **the CAP files, the bytecode and its data argument,** when shared between the TSF and another trusted IT product, the TSF shall use: • The rules defined in [JCVM304] specification; • The API tokens defined in the export files of reference implementation • **The rules defined in ISO 7816-6** • **The rules defined in [GP221] specification**	FPT_TDC.1
The TSF shall preserve a secure state when the following types of failures occur: **those associated to the potential security violations described in FAU_ARP.1.** The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE304], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE304] §4.1.2). Behavior of the TOE on power loss and reset is described in [JCRE304], §3.6, and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE304], §3.6.2	FPT_FLS.1/JCS
Only authorized users can observe the operation **cryptographic operations / comparisons operations** on **Key values / PIN values** by **S.JCRE, S.Applet.**	FPR_UNO.1

9.1.1.4 SF.AID: AID Management

Only **the JCRE** can **modify** the **list of registered applets' AIDs**.	FMT_MTD.1/JCRE
Only secure values are accepted for **the AIDs of registered applets**.	FMT_MTD.3/JCRE
The TSF shall maintain the following list of security attributes belonging to individual users: • **package AID**	FIA_ATD.1/AID

• **Applet's version number** • **registered applet's AID** • **applet selection status ([JCVM304], §6.5)**	
The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.	FIA_UID.2/AID
Initial applet selection is performed as described in [JCRE304]§4 **Applet selection is performed after a successful SELECT FILE command as described in [JCRE304]§4**.	FIA_USB.1/AID

9.1.1.5 SF.INST: Installer

the protocol used provides for the unambiguous association between the security attributes and the user data received: The format of the CAP file is precisely defined in Sun's specification ([JCVM304]); it contains the user data (like applet's code and data) and the security attribute altogether.	FDP_ITC.2/Installer
Each package contains a package Version attribute, which is a pair of major and minor version numbers ([JCVM304], §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 ([JCVM304], §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 compatibles. However, package files do have "package Version Numbers" ([JCVM304]) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.	FDP_ITC.2/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 loading is allowed only if, for each dependent package, its AID attribute is equal to a resident package AID attribute, the major (minor) Version attribute associated to the former is equal (less than or equal) to the major (minor) Version attribute associated to the latter ([JCVM304],§4.5.2).	FDP_ITC.2/Installer
The TSF shall maintain the roles**: the installer**	FMT_SMR.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 [JCRE304] §11.1.4**	FPT_FLS.1/Installer
After **Failure during applet loading, installation and deletion; sensitive data loading**, the TSF ensures the return of the TOE to a secure state using automated procedures. The TSF provides the capability to determine the objects that were or were not capable of being recovered.	FPT_RCV.3/Installer

9.1.1.6 SF.ADEL: Applet Deletion

Only **the Java Card RE (S.JCRE)** can **modify** the security attributes: **ActiveApplets**. The modification of the ActiveApplets security attribute should be performed in accordance with the rules given in [JCRE304], §4.	FMT_MSA.1/ADEL
Provide restrictive default values for security attributes that are used to enforce the SFP.	FMT_MSA.3/ADEL
The TSF shall maintain the roles: **the applet deletion manager**.	FMT_SMR.1/ADEL
The TSF shall be able to **Modify the ActiveApplets security attribute**.	FMT_SMF.1/ADEL

([JCRE304], §11.3.4.1, Applet Instance Deletion). The S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if, (1) S.ADEL is currently selected, (2) O.APPLET is deselected and (3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE304], §8.5) O.JAVAOBJECT is remote reachable.	FDP_ACC.2/ADEL FDP_ACF.1/ADEL
([JCRE304], §11.3.4.1, Multiple Applet Instance Deletion). The S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if, (1) S.ADEL is currently selected, (2) every O.APPLET being deleted is deselected and (3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE304], §8.5) O.JAVAOBJECT is remote reachable.	FDP_ACC.2/ADEL FDP_ACF.1/ADEL
([JCRE304], §11.3.4.2, Applet/Library Package Deletion). The S.ADEL may perform OP.DELETE_PCKG upon an O.CODE_PCKG only if, (1) S.ADEL is currently selected, (2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PCKG that is an instance of a class that belongs to O.CODE_PCKG exists on the card and (3) there is no package loaded on the card that depends on O.CODE_PCKG.	FDP_ACC.2/ADEL FDP_ACF.1/ADEL
([JCRE304], §11.3.4.3, Applet Package and Contained Instances Deletion). The S.ADEL may perform OP.DELETE_PCKG_APPLET upon an O.CODE_PCKG only if, (1) S.ADEL is currently selected, (2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PCKG, which is an instance of a class that belongs to O.CODE_PCKG exists on the card, (3) there is no package loaded on the card that depends on O.CODE_PCKG and (4) for every O.APPLET of those being deleted it holds that: (i) O.APPLET is deselected and (ii) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted, or ([JCRE304],§8.5) O.JAVAOBJECT is remote reachable.	FDP_ACC.2/ADEL FDP_ACF.1/ADEL
However, the S.ADEL may be granted privileges ([JCRE304], §11.3.5) to bypass the preceding policies. For instance, the logical deletion of an applet renders it un-selectable; this has implications on the management of the associated TSF data (see application note of FMT_MTD.1.1/JCRE).	FDP_ACF.1/ADEL
Only the S.ADEL can delete O.CODE_PKG or O.APPLET from the card.	FDP_ACF.1/ADEL
Upon deallocation of a resource from the **applet instances and/or packages when one of the deletion operations in FDP_ACC.2.1/ADEL is performed on them**, any previous information content of the resource is made unavailable.	FDP_RIP.1/ADEL
Requirements on de-allocation during applet/package deletion are described in [JCRE304], §11.3.4.1, §11.3.4.2 and §11.3.4.3.	FDP_RIP.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 [JCRE304], §11.3.4**.	FPT_FLS.1/ADEL

9.1.1.7 SF.ODEL: Object Deletion

Upon deallocation of the resource from the objects owned by the context of an applet instance which triggered the execution of the method javacard.framework.JCSystem.requestObjectDeletion(), any previous information content of the resource is made unavailable.	FDP_RIP.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.	FPT_FLS.1/ODEL

9.1.1.8 SF.CAR: Secure Carrier

No one can **modify** the security attributes **AID**	FMT_MSA.1/CM
Default values for security attributes are: • User role: none • Applet checked: No • DAP Key OK: No	FMT_MSA.3/CM
The TSF shall maintain the roles**: Card Manager**	FMT_SMR.1/CM
The Card Manager loads applets with their AID.	FMT_SMF.1/CM
The TOE enforces the generation of evidence of origin for transmitted **application packages** at all times.	FCO_NRO.2/CM
The TOE allows: • JCAPI with already installed applets • APDUs for Applets on behalf of the user to be performed before the user is authenticated.	FIA_UAU.1/CM
The TOE allows: • JCAPI with already installed applets • APDUs for Applets on behalf of the user to be performed before the user is identified.	FIA_UID.1/CM
• Only the user with the security attribute role set to Operator can load an applet. • Only applets with the security attribute Checked set to YES can be transferred. • The DAP key OK security attribute must be set to TRUE to check the integrity and the origin of the applet	FDP_IFC.2/CM FDP_IFF.1/CM
Package loading is protected against **modification, deletion, insertion, and replay** errors. If such an error occurs, it is detected at reception .	FDP_UIT.1/CM
New packages can be loaded and installed on the card only on demand of the card issuer. This is done through a GP Secure Channel.	FTP_ITC.1/CM

9.1.1.9 SF.SCP: Smart Card Platform

The TSF periodically tests the security mechanisms of the IC. It also checks the integrity of sensitive assets: Applets, PIN and Keys.	FPT_TST.1/SCP
The TSF resists physical attacks	FPT_PHP.3/SCP
The TSF offers transaction mechanisms	FPT_RCV.4/SCP

9.1.1.10 SF.CMG: Card Manager

The Card Manager loads and extradites applets. It also loads GP key.	FDP_ACC.1/CMGR FDP_ACF.1/CMGR

No one can modify the security attribute code category	FMT_MSA.1/CMGR
Only restrictive default values can be used for the code category	FMT_MSA.3/CMGR

9.1.1.11 SF.SPAPI: Specific API

Provides means to application to control execution flow, to detect any failure and to react if required	FPT_FLS.1/SpecificAPI
Provides means to application to execute securely data transfer and comparison, to detect any failure during operation and to react if required..	FPT_ITT.1/SpecificAPI
Provides means to introduce dummy operations leading to unobservability of sensitive operation	FPR_UNO.1/SpecificAPI

9.1.1.12 SF.RND: RNG

Provide a random value	FCS_RND.1

9.1.2 TSFs provided by the CENTAURUS IC

The evaluation is a composite evaluation and uses the results of the CC evaluation provided by [CR-IC]. The IC and its primary embedded software have been evaluated at level EAL 5+. These SF are the same for the IC considered in this ST, CENTAURUS FB 04.

SF	Description
SF_PMODE	Product Mode
SF_IDENT	Identification
SF_CONF&INT	Confidentiality and integrity
SF_SCRA	Scrambling
SF_EXEC	Correct Execution
SF_EM	Environment Control
SF_ALARM	Alarm Management
SF_RANDOM	Randomization
SF_RNG	Random Number Generator
SF_DE	Design
SF_LOAD	Loader
SF_CRYPTO	Cryptographic Services

Table 11: Security Functions provided by the Invia CENTAURUS FB 04 chip

These SF are described in [ST-IC].

9.2 ASSURANCE MEASURES

Assurance Measure	Document title
AM_ASE	MultiApp Essential V1.1 JCS Security Target
AM_ADV_Spec	Functional Specifications - MultiApp Essential V1.1
AM_ADV_Design	Design – MultiApp Essential V1.1
AM_ADV_Int	Internals – MultiApp Essential V1.1
AM_ALC	Class ALC – MultiApp Essential V1.1
AM_AGD	Guidance – MultiApp Essential V1.1
AM_ATE	Class ATE – MultiApp Essential V1.1

AM_CODE	Source Code – MultiApp Essential V1.1
AM_Samples	Samples – MultiApp Essential V1.1

Table 12: Assurance Measures.

The development team uses a configuration management system that supports the generation of the TOE. The configuration management system is well documented and identifies all different configuration items. The configuration management tracks the implementation representation, design documentation, test documentation, guidance documentation. The security of the configuration management is described in detail in a separate document.
The delivery process of the TOE is well defined and follows strict procedures. Several measures prevent the modification of the TOE based on the developer's master copy and the user's version. The Administrator and the User are provided with necessary documentation for initialization and start-up of the TOE.
The implementation is based on an informal design of the components of the TOE. The description is sufficient to generate the TOE without other design requirements.
The correspondence of the Security Functional Requirements (SFR) with less abstract representations will be demonstrated in a separate document. This addresses ADV_ARC, ADV_FSP, ADV_IMP, and ADV_TDS.
The tools used in the development environment are appropriate to protect the confidentiality and integrity of the TOE design and implementation. The development is controlled by a life cycle model of the TOE. The development tools are well defined and documented.
The Gemalto R&D organization is equipped with organizational and personnel means that are necessary to develop the TOE.
As the evaluation is identified as a composite evaluation based on the CC evaluation of the hardware, the assurance measures related to the hardware (IC) will be provided by documents of the IC manufacturer.

10 RATIONALES

10.1 PP CLAIMS RATIONALE

This Security Target is conformant with the Protection Profile "Java Card System, Open configuration", [PP-JCS-Open]. The Open 3 Classic Edition configuration of [PP-JCS-Open] is used.

As the Card Manager is included in the TOE, OE.CARD-MANAGEMENT is changed into the following Objective for the TOE: O.CARD-MANAGEMENT.

As the SCP is included in the TOE, OE.SCP.RECOVERY, OE.SCP.SUPPORT, and OE.SCP.IC are changed into the following Objectives on the TOE: O.SCP.RECOVERY, O.SCP.SUPPORT, and O.SCP.IC.
There are extra TOE objectives, O.SpecificAPI and O.RND. to provide additional services to applications. Such extension has no impact on PP coverage.
As no other modification was done, we can conclude that the conformance is demonstrated

END OF DOCUMENT