MORPHO

SECURITY TARGET LITE
IDEAL CITIZ V2.1 OPEN PLATFORM

Reference: 2015_2000009094

DOCUMENT EVOLUTION

Date	Index	Author	Revision
01/09/2015	01	MORPHO	Initial Version
08/09/2015	02	MORPHO	Update

Table of contents

1.1 Table of figures

1.2 TOE reference

The Target Of Evaluation (TOE) is the open smartcard platform developed by Morpho, identified as "*IDeal Citiz v2.1 open platform".* The TOE offers GlobalPlatform compliant card management capabilities in conjunction with an open JavaCard platform. The system is based on native operating system services and embedded in a smartcard on top of the chip M7892 B11 with optional RSA2048/4096 v1.02.013, EC v1.02.013, SHA-2 v1.01 and Toolbox v1.02.013 libraries and with specific IC dedicated software (firmware) produced by Infineon Technologies AG.

TOE Reference	
Product:	*IDeal Citiz v2.1 open platform*
Version number:	2.1.0
Date of release	20/03/2015
TOE Identification Method	See [AGD_PRE]
Origin:	Morpho
Chip Identifier:	Infineon M7892 B11 with optional RSA2048/4096 v1.02.013, EC v1.02.013, SHA-2 v1.01 and Toolbox v1.02.013 libraries and with specific IC dedicated software (firmware)
Chip Ref. Certificate	M7892 : BSI-DSZ-CC-0782-2012-MA-01

This security target states the security requirements that are met by the TOE, provides an overview on the security functionality offered by the product and describes the intended usage of the TOE.

1.3 Security Target Identification

Security Target identification is described in the table below:

ST Identification	
Title:	Security target LITE Ideal Citiz v2.1 open platform
Version:	*02*
Reference:	2015_2000009094
Origin:	Morpho
Product Identification:	*IDeal Citiz v2.1 open platform*
Chip Identifier:	Infineon M7892 B11 with optional RSA2048/4096 v1.02.013, EC v1.02.013, SHA-2 v1.01 and Toolbox v1.02.013 libraries and with specific IC dedicated software (firmware)
Chip Ref. Certificate	M7892 : BSI-DSZ-CC-0782-2012-MA-01
Assurance Level:	EAL5 augmented with ALC_DVS.2 and AVA_VAN.5
CC Version:	3.1 Release 4
Compliant Protection Profile	JavaCard System – Open Configuration - Protection Profile, Version 3.0, May 2012. Certified by ANSSI under the reference ANSSI-CC-PP-2010/03-M01

1.4 TOE documentation

TOE documentation is described in the table below:

TOE documentation	
[AGD_PRE]	2014_2000003597 – PRE – IdealCitiz_v2_1 – Preparative Procedures, version 06 – 25/03/2015
[AGD_OPE]	2014_200004459 - OPE - IDealCitiz_v2_1 - Operational User Guidance, Version 02 – 24/04/2014
[API]	2015_2000010613 - IDealCitiz v2_1 - Global Platform and JavaCard API, Version 01
[VAR]	2014_2000001513 - IDeal Citiz v2.1 – VerificationAuthorityRules, Version 1.0
[BADR]	2014_2000001499 - IDeal Citiz v2.1 – BasicAppletDevelopmentRecommendations, Version 1.0
[SADR]	2014_2000001501 - IDeal Citiz v2.1 – SecureAppletDevelopmentRecommendations, Version 1.2

2 TOE overview

The TOE is an open operating system platform primarily intended to host a set of core native applications (high level of complexity supporting security evaluations) and a JavaCard platform for customer or third party applet development. For that, the system can be augmented with additional applets installed during the pre-issuance and/or the post-issuance phase of TOE's life cycle.

The TOE is a high-security system evaluated according to the CC assurance level EAL5 augmented with ALC_DVS.2 and AVA_VAN.5 offering strong security services to the application layer.

2.1 TOE type

The TOE is composed of the smartcard integrated circuit Infineon (IFX) M7892 B11 containing the MOS ID JC operating system as embedded software. The operating system implements GlobalPlatform services and an open JavaCard platform. The functional level of the operating system is based on a Java multi-application platform, compliant with GlobalPlatform 2.1.1 specifications and JavaCard 3.0.1 classic edition, May 2009, including Specification Errata, October 2010, Updated February 2011.

IDeal Citiz v2.1 open platform is an Identity product suitable to the Identity Market worldwide thanks to its Common Criteria certifications.

It contributes to the move into the digital world by providing secured and convenient online government services to the citizens: strong authentication solution ensuring privacy and ease of use.

It is an anti-fraud tool to protect against usage of fake identity on the field and on the internet thus allowing cost savings and ROI. The identification on the web could be performed towards multiple access devices.

One product securelly houses numerous applications and creates a convenient electronic document for the user and the issuers, as well as cost-effectivness by avoiding the creation of several single-purpose documents.

2.2 Usage and major security features of the TOE

The TOE offers the following major product features:

- the contact interface protocol according to ISO or EMV standards (mutually exclusive)
- the contactless interface compliant with ISO 14443 (Type A)
- JavaCard V3.0.1 (classic) compliant Java Platform Implementation
- GlobalPlatform
    - Delegated Management
    - Security Domain with DAP verification (PKCS scheme) and Delegated Management
    - Manage CVM (Application Privilege)
    - Application Extradition.

The major security features of the TOE are the following:

- Secured GlobalPlatform Card Management and GlobalPlatform Domain Separation which allows for a protected post-issuance applet installation under full control of the card issuer, mobile network operator, and application provider respectively
- Protected JavaCard Virtual Machine and Runtime Environment, including Strong JavaCard Applet Isolation, to protect the integrity and confidentiality of sensitive applet data
- Strong cryptographic support including

  - (T)DES cipher
  - AES cipher with up to 256 bits key length
  - RSA with up to 3072 bits key length
  - ECC with up to 521 bits key length
- Secure data personalization through GlobalPlatform Secure Channel Protocol:
  - SCP02 (i=15, i=35)
  - SCP03 supported according to GlobalPlatform 2.2 Amendment D.

The TOE allows for post-issuance download of additional applets by using Global Platform mechanisms to add value added services to the card and to provide a maximum level of the configurability.

In the same time, the strong protection mechanisms ensure that applet data and code are isolated and that the overall integrity of the system is always protected. Thus, the platform is able to host sensitive applets like mobile payment or transport applications.

2.3 Required non-TOE hardware/software/firmware

The TOE requires the following non-TOE hardware, software and firmware.

2.3.1 Off-Card Bytecode Verifier

The TOE, and in particular the underlying JavaCard Platform rely on an off-card 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.

2.3.2 Contact Based Communication

For direct contact-based communication the environment uses the ISO7816 contact plate of the TOE. Therefore, no specific additional hardware is required by the TOE itself.

2.3.3 Contactless Communication

The TOE establishes contactless communication to a card acceptance device directly using the antenna embedded in the plastic body.

2.3.4 Software Components out of TOE Scope

All native parts of the MOSID system and also pre-loaded applet mechanism is considered in the evaluation even though some components are "SFR-non-interfering".

For a detailed overview of the precise TOE boundary and the separation into SFR-related and SFR-non-interfering parts refer to the TOE description in the next chapter.

2.4 Actors of the TOE

One of the characteristics of the JavaCard platforms is that several entities are represented inside these platforms:

- The **Application Provider** (AP), entity or institution responsible for the applications and their associated services. It is mainly a government institution.
- The **Card Issuer** (CI), entity or institution responsible for the Card issuance and administration. It is mainly the Government.

3 TOE description

3.1 Physical scope of the TOE

From a physical point of view, the Target of Evaluation (TOE) consists of those hardware and software resources of an IC with embedded software. All non-IC components of the smart card (e.g. magnetic stripes, holograms, printed or embossed data...) are outside TOE perimeter.

From the physical point of view the Target of Evaluation (TOE) is a Smartcard consisting of the MOS ID operating system embedded into an IXF M7892 B11 Smartcard integrated circuit. For details refer to the description of the logical TOE scope in the next section.

The product is a smart card which uses the following pins as described in the Figure 1 below for communication.

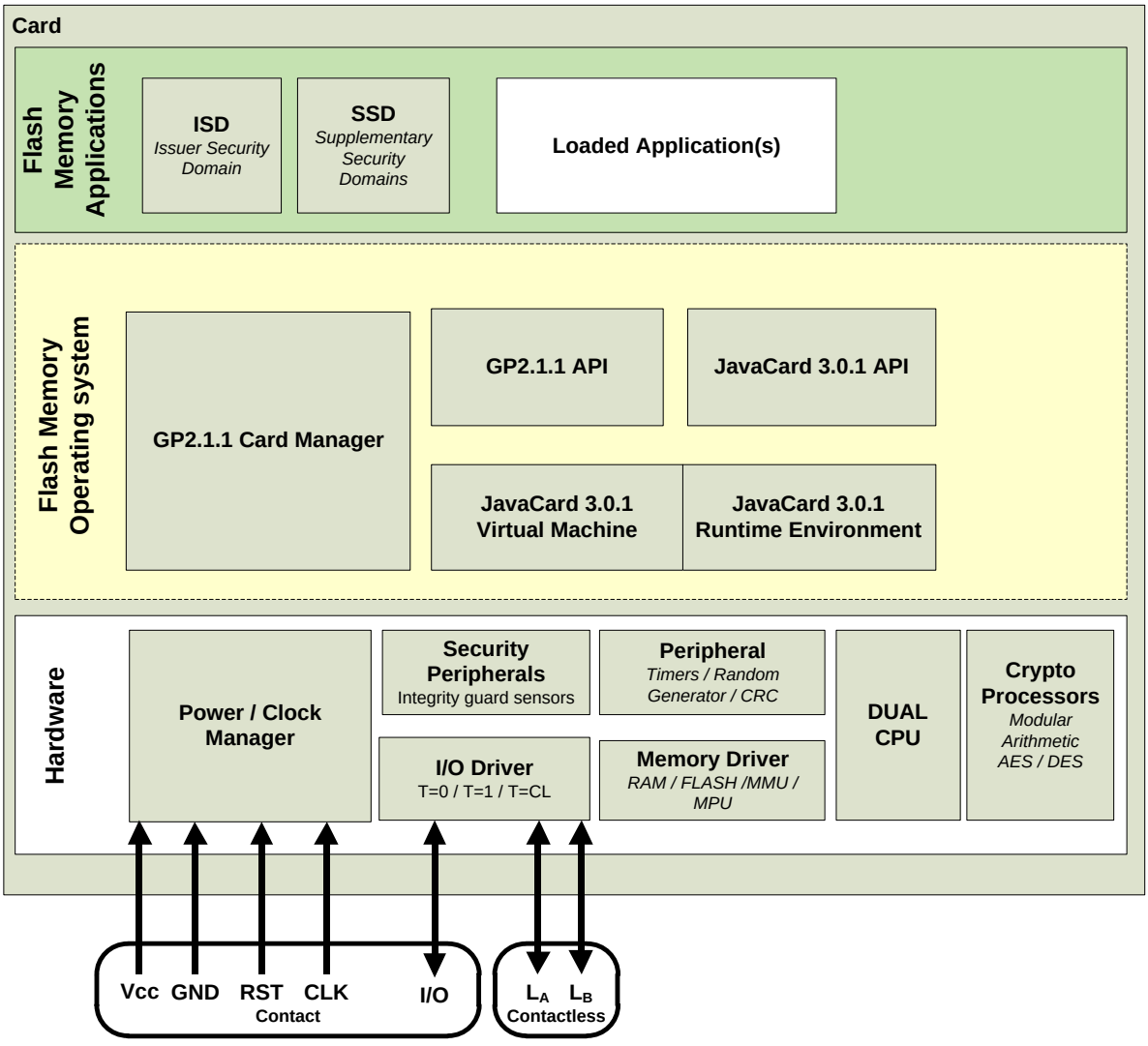

Figure 1 – View of the smartcard and pins

3.2 Logical scope of the TOE

The figure 2 below shows the different elements that compose the card. The hardware platform is the M7892 B11 security IC manufactured by Infineon.

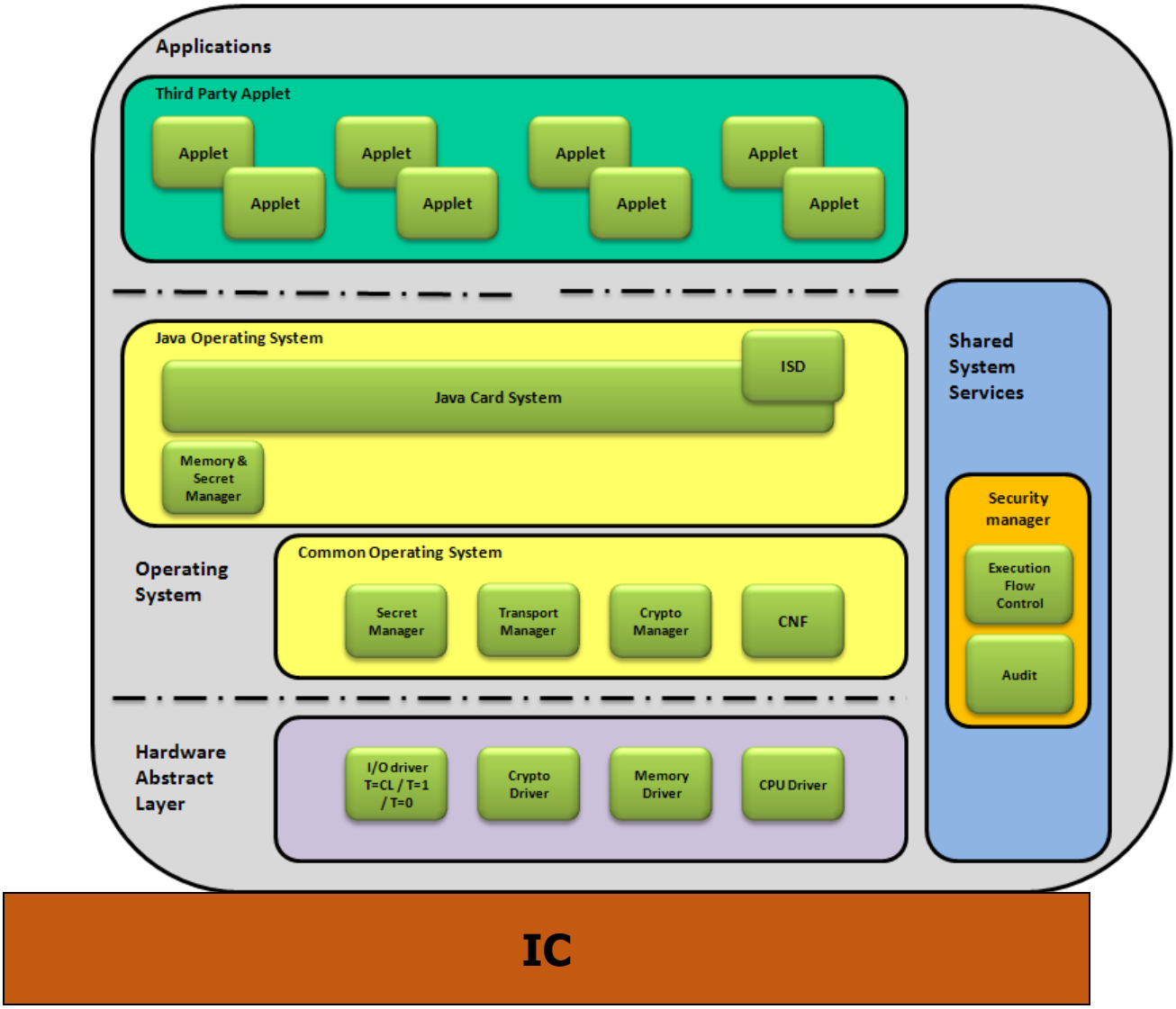

Figure 2 Product architecture

The MOS ID operating system is a full operating system implementing an open JavaCard platform and GP card management.

With respect to the TOE scope the following aspects have to be considered:

- the product evaluation is a composite evaluation which re-uses the result of a baseline certification conducted by the IC manufacturer. This baseline evaluation covers the dark-orange-coloured components in the product, namely the integrated circuit IC and its Crypto Library.

- the security functionality implementing core of the system consist of the common operating system, the JavaCard system and the secured card content management functions as specified by Global Platform. Additionally, the security domain hierarchy which is part of a customer profile has to be taken into account.

3.2.1 IC, Crypto Library

The M7892 B11 consists of Security Dual Interface Controllers as integrated circuits, meeting the highest requirements in terms of performance and security. They are manufactured by Infineon Technologies AG in a 90 nm CMOS-technology (L90). This M7892 B11 consists of a core system, memories, co-processors, peripherals, security modules and analogue peripherals. The major components of the core system are the two CPUs (Central Processing Units), the MMU (Memory Management Unit) and MED (Memory Encryption/Decryption Unit). The co-processor block contains the processors for RSA/ECC and DES/AES processing.

The chip contains two co-processors for cryptographic operations: The Crypto2304T for calculation of asymmetric algorithms like RSA and Elliptic Curve (ECC) and the Symmetric Cryptographic Processor (SCP) for dual-key or triple-key triple-DES and AES calculations.

3.2.2 Common Operating System

The Common Operating System is the inner core of the MOS ID. Its primary purpose is to implement an interface between the IC with crypto library and the JavaCard System. It serves as a hardware abstraction layer and implements the following functionalities.

- APDU I/O management
- Memory access and management
- Transaction management
- Exception management
- Timer management
- (Interface to hardware / library) cryptographic functions
- Chip bootstrap
- Chip initialisation

3.2.3 JavaCard System (JCS)

The JCS allows JavaCard based applications (applets) to be run securely on smart cards. The JCS consists of the JavaCard virtual machine (JCVM), the JavaCard runtime environment (JCRE) and the JavaCard Application Programming Interface (JC API). The JavaCard provides a layer between a native platform and an applet space. That layer allows applications written for one smart card platform enabled with JavaCard technology to run on any other such platform.

The JCVM is essentially an abstract machine that specifies the behaviour of the bytecode interpreter embedded in the card, controls memory allocation, manages objects, and enforces the runtime security.

- The native methods provide support to the JCVM and the next-layer system classes. They are responsible for handling the low-level communication protocols, memory management, cryptographic support etc.
- The system classes are in charge of managing transactions, managing communication between the host applications and JC applets, and controlling applet creation, selection, and de-selection.

- The API provides classes and interfaces for the core functionality of JavaCard applications. It defines the calling conventions by which an applet may access the JC and native services such as, I/O management functions, PIN and cryptographic specific management and the exceptions mechanism.

The JCS is based on JavaCard 3.0.1 classic edition; see **[JCVM]**, **[JCRE]** and **[JCAPI]**.

3.2.4 Card Management (CM)

The Card Management component of the TOE implements the functionalities specified in **[GP_CS]**. These functionalities provide APIs and technologies for secure management of the card content and especially for the applications hosted by the card.

The Card Management component includes the following optional functionalities:

- Logical Channel Management (channel 1, 2 and 3)
- Supplementary Security Domains
- SCP02
- SCP03
- Delegated Management (without blacklist support)
- DAP Verification and Mandated DAP
- RSA key support (1024 bits)
- Cardholder Verification Method (CVM)
- Global Services Management
- GET STATUS, completely supported (incl. tag list)

3.2.5 Cryptographic algorithms and functionality

- 3DES (56, 112 and 168 bit keys) for en-/decryption (CBC and ECB) and MAC generation and verification (Retail-MAC, CMAC and CBC-MAC)
- AES (Advanced Encryption Standard) with key length of 128, 192, and 256 Bit for en-/decryption (CBC and ECB) and MAC generation and verification (CMAC, CBC-MAC)
- RSA (768 up to 2112 bits keys) and RSA CRT (768 up to 3072 bits keys) for en-/decryption and signature generation and verification.
- RSA and RSA CRT key generation (1976 up to 3072 bits keys)
- SHA-1, SHA-224, SHA-256, SHA-384 and SHA-512 hash algorithm.
- ECC algorithm that can be used for signature generation and signature verification (ECDSA) from 128 to 521 bits.
- ECC key generation algorithm that can be used to generate ECC key pairs.
- Random number generation

3.2.6 PACE API

The PACE API is part of the TOE and provides the following services:

- SAC PACE authentication (DES/AES algorithms)

- Secure Messaging initialization with session keys issued from the PACE authentication.
- PACE mapping (point generation with ECDH and domain generation)

PACE API is aimed at providing a replacement for the BAC protocol, correcting the entropy weakness with strong session keys. This API provides services as secure messaging and mapping.

This API is optional as it can be removed during personalisation phase: The memory zone which contains data for PACE is made unavailable and error messages are returned when API is called.

3.2.7 Life cycle

The following description introduces generics but fine-grained 3 models for the life-cycle of secure smartcard products. The 3 models are compliant to standard smartcard life-cycle models as defined e.g. in **[PP_JC]** and **[PP_IC]**.

The intent of the more fine-grained models is to cover the specific aspects of new technologies like flash or applet loading in a comprehensive way and to add some flexibility with respect to the separation of responsibilities between the various parties involved. Consider the following 3 life-cycle supported for this product (LC1 to LC3).

The smartcard product life-cycle is decomposed in 7 steps that describe the competent authorities for each of these steps. The purpose of the embedded software designed in step 1 is to control and protect the TOE during steps 4 to 7 (product usage).

The embedded software development is the core scope of the composite evaluation and corresponds directly to step 1 of the standard smartcard life-cycle defined in **[PP_IC]**.

The step 2 "IC Development" of the standard life-cycle is directly visible in the life-cycle overview.

The Life Cycle model extracts the OS Flash-loading process from step 3 "IC Manufacturing" because for Flash products the "OS Loading" is no longer directly coupled to the IC manufacturing. Thus, the IC manufacturing primarily deals with the physical manufacturing of the IC (production of wafers) and IC pre-personalisation. Whether the IC manufacturer takes also care of the OS loading (either by masking or by flashing) depends on the product and is detailed in the concrete product-type specific instantiation of the life-cycle.

The step 4 "IC initialization" from the standard model is also focussed on the logical production steps which are detailed into "OS loading" (masking or flashing). During this step the Morpho software is loaded with a blackbox. The generic blackbox contains only authentication keys. The dedicated blackbox contains keys and specific initialization and personalization values. Moreover, this step includes the configuration (CNF) stage, which is a Morpho proprietary card life cycle stage.

The step 5 "Product Pre-Personalisation" corresponds to the loading of non-card individual data.

The step 6 "Personalisation" of the standard model corresponds directly to the "Product Personalisation" (loading of card individual data). During the personalisation, the PACE API can be removed, as described in 3.2.5

3.2.7.1 Life Cycle 1

For this Life Cycle "LC1", the wafer is manufactured and initialized at the founder site. It is then shipped to Morpho, successively through the module and the inlay (CL interface only) manufacturers. Morpho is responsible for the embedding process and the pre-personalization of the card. Finally, the card is shipped to the Personalizer. During the shipment from IC manufacturer to Morpho the chip is protected by a diversified key derived from the Morpho

factory dedicated master key. During the shipment from Morpho to the Personalizer, the card is protected by a diversified key derived from a Personalizer dedicated master key.

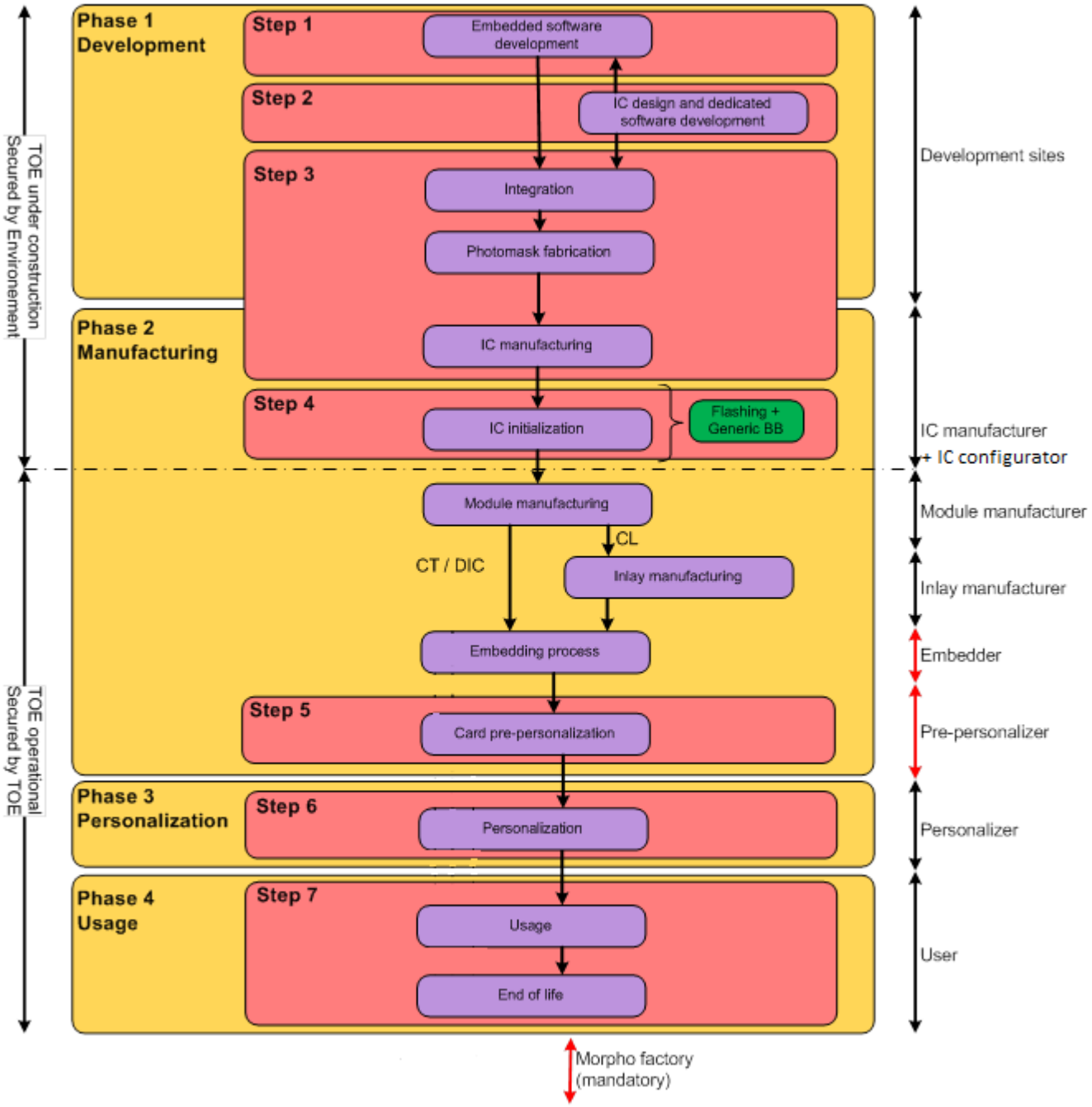

Figure 3 Flashing and generic BlackBox loading at founder site on wafer; pre-personalization on card

3.2.7.2 Life Cycle 2

For this Life Cycle "LC2", the wafer is manufactured and initialized at the founder site. It is then shipped to Morpho through the module manufacturer. Morpho is responsible for pre-personalization of the module prior being

sent successively to the inlay manufacturer and the embedder. Finally, the card is shipped to the Personalizer directly. During the shipment from IC manufacturer to Morpho, the chip is protected by a diversified key derived from the Morpho factory dedicated master key. During the shipment from Morpho to the Personalizer (through the inlay manufacturer and the embedded), the chip is protected by a diversified key derived from a Personalizer dedicated master key.

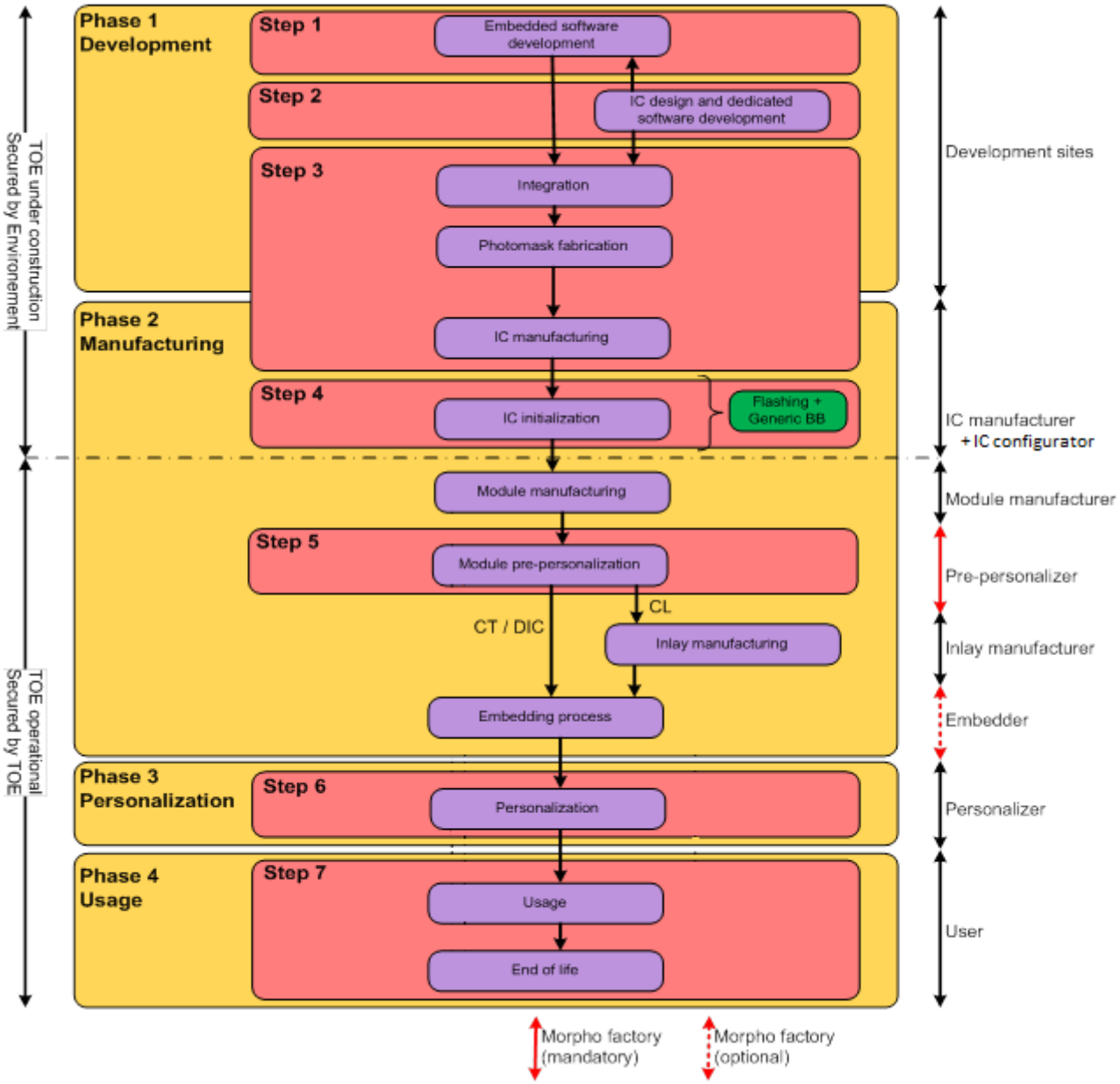

Figure 4 Flashing and generic BlackBox loading at founder site on wafer; pre-personalization on module

3.2.7.3 Life Cycle 3

For this Life Cycle "LC3", the wafer is manufactured and initialized at the founder site. It is then shipped to the Pre-personalizer successively through the module, the inlay (CL interface only) manufacturers and the embedder.

Finally, the card is shipped to the Personalizer directly. During the shipment from IC manufacturer to the Pre-personalizer, the chip is protected by a diversified key derived from Pre-personalizer dedicated master key. During the shipment from the Pre-personalizer to the Personalizer, the card is protected by a diversified key derived from a Personalizer dedicated master key.

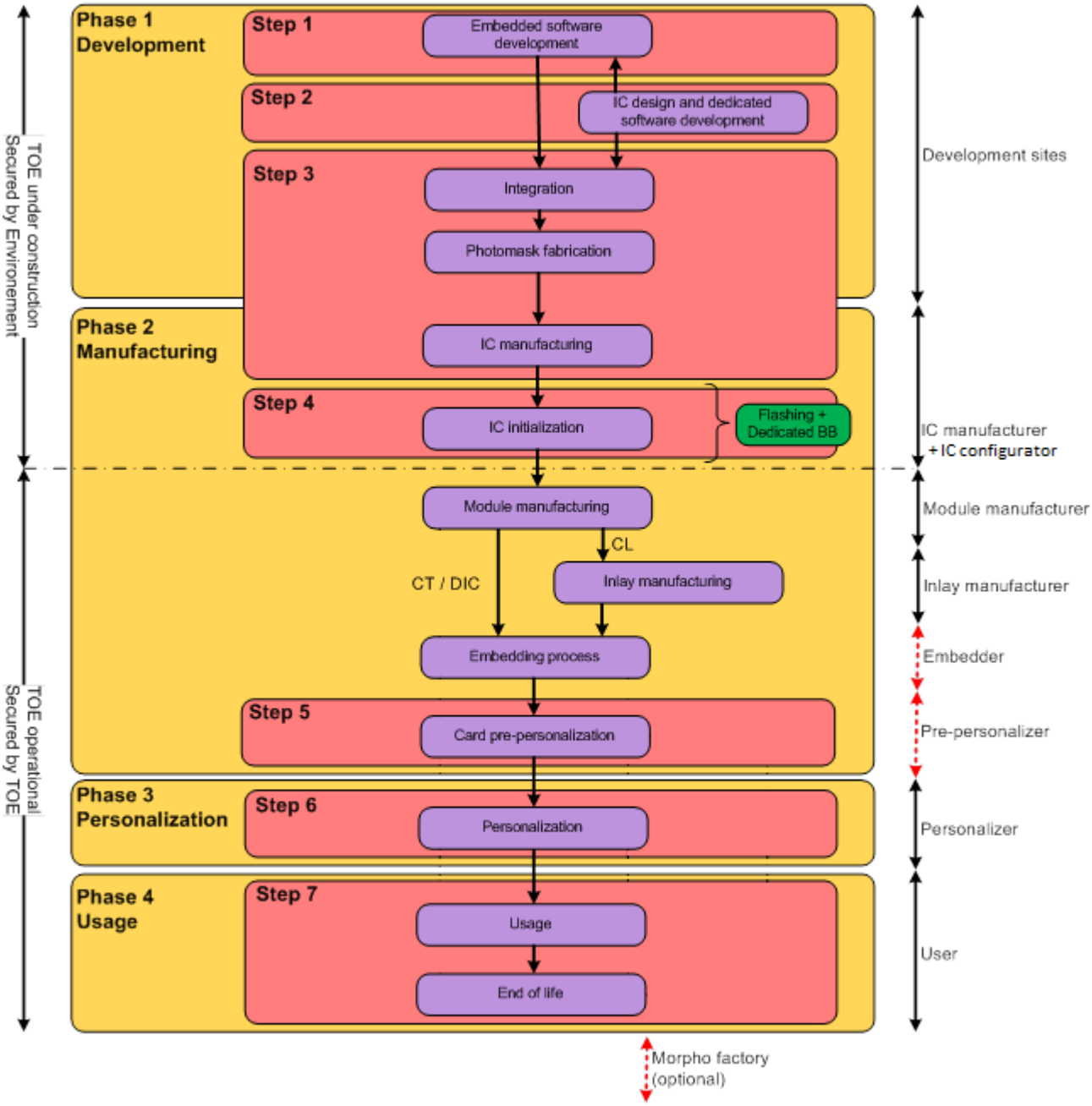

Figure 5 Flashing and dedicated BlackBox loading at founder site on wafer; pre-personalization on card

3.2.7.4 Actors

The actors of the smart card product life-cycle are listed on the table below.

Actors		Identification
Integrated Circuit (IC) developer		Infineon
Embedded software developer		Morpho
Card manufacturer	Integrated Circuit (IC) manufacturer	Infineon
	Integrated Circuit (IC) configurator	Morpho
	Module manufacturer	Nedcard, Infineon, Morpho, customer defined
	Pre-personalizer	Morpho or customer defined
	Inlay manufacturer	KnL, SMT, HID, PAV, I2PL, SPS, ASK, customer defined, etc
	Embedder	Morpho or customer defined
Personalizer		The agent who is acting on the behalf of the issuing State or Organization and personalize the card for the holder by activities establishing the identity of the holder with biographic data.
Card Holder / User / Issuer		The rightful holder of the card for whom the issuing State or Organization personalizes the card.

Table 1: Actors of the smart card product

3.2.7.5 Description of the TOE environment

The TOE environment is defined as follows:

- Development environment corresponding to steps 1 and 2;
- Production environment:
  - IC Photomask fabrication and IC Manufacturing environment corresponding to steps 3;
  - Smartcard finishing process environment and pre-personalisation (initialization) corresponding to steps 4 and 5;
  - Personalization environment corresponding to step 6.
- Card exploitation environment corresponding to step 7.

4 Common Criteria Conformance Claim

4.1 Common Criteria Conformance

This Security Target claims conformance to Common Criteria version 3.1 Revision 4, with the following documents:

- "Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model", **[CC_Part1]**
- "Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional requirements", **[CC_Part2]**
- "Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance requirements", **[CC_Part3]**

Conformance is claimed as follows:

- Part 1: conformant
- Part 2: conformant
- Part 3: conformant, compliant to EAL5 augmented with ALC_DVS.2 and AVA_VAN.5.

4.2 Conformance with an assurance package

The set of assurance requirements is the package EAL5 augmented by:

- ALC_DVS.2, "*Sufficiency of security measures*"
- AVA_VAN.5, "*Advanced methodical vulnerability analysis*"

Assurance requirements are split in two packages, one for the TOE itself and one for its development environment, allowing for separate package assessment. However, both assessments must be combined in order to fulfill the whole set of PP assurance requirements.

4.2.1 AVA_VAN.5 and Industrial Key Length Requirements

The assurance level EAL5 augmented with AVA_VAN.5 implies that the product shall enforce resistance against an attacker with a high-attack potential. This in particular also has implications on brute-force attacks on cryptographic algorithms. With respect to such kinds of attacks, the resistance of a cryptographic scheme depends massively on the length of the cryptographic key material used, which defines the size of the key space a single key is selected from. Therefore, certification bodies and other national cryptography approval bodies publish minimum requirements for the key material of the various crypto schemes.

For example, the currently recommended length for RSA keys is 2048 bits and the long-term resistance of TDES is no longer confirmed.

However, some industrial standards still use shorter key lengths than those considered the recommended minimum for ensuring resistance against brute force attacks with a high attack potential. For example, the usage of RSA keys in the context of GlobalPlatform card content management permits the usage of 1024 bits RSA keys.

This usage of slightly weaker keys than the recommended ones can be perfectly reasonable from a risk management perspective of the customer. This is in particular true if a long-term migration strategy is to be taken into account for updating an already running eco-system including e.g. a large background infrastructure. However, the usage of no longer recommended key length produces a formal incompliance with the AVA_VAN.5 resistance level the product has to achieve.

To resolve this issue, the assurance claim should be interpreted more precisely as follows for the product specified in this security target: the product reaches the assurance level EAL5 augmented with AVA_VAN.5 if appropriate, recommended key length are used. Even in the case that industrial standards require the usage of weaker keys, the product is still highly-resistant with respect to all attack scenarios with the limitation of highly resistant to all attack scenarios with the limitation of brute force attacks on small keys.

4.3 Conformance with a Protection Profile

This Security Target is conformant with the JavaCard Protection Profile **[PP_JC]**, without the RMI (Remote Method Invocation) option, which is not implemented and out of the scope of this evaluation. Therefore RMI related entities, subject, object, information, operation and security attribute, as well as the corresponding SFRs of the JavaCard Protection Profile **[PP_JC]** are excluded from this Security Target.

4.4 Conformance Rationale

This Security Target claims a Strict conformance with the JavaCard Protection Profile **[PP_JC]**.

4.4.1 TOE type consitency

The TOE type is "JavaCard 3.0.1 conformant to GlobalPlatform 2.1.1, implemented on a chip Infineon of the M7892 B11 family" and protection profile TOE type is "smart card platform enabled with JavaCard technology". TOE types are compatible since the security target's TOE is a smart card that is enabled with JavaCard technology.

4.4.2 SPD statement consitency

All assets and threats from **[PP_JC]** are included in this ST. The only exception is that the threat T.EXE-CODE-REMOTE is removed due to the fact that the product does not support the threatened RMI functionality.
4 additional assets have been added to those of the **[PP_JC]** : D.COMMAND, D.GP_CODE, D.SD_KEYS, D.ISD_KEYS. These assets have been included because the security domain and the GlobalPlatform framework are parts of the TOE.
1 optional asset from the **[PP_JC]** , D.BIO, has been included because biometric templates are part of the TOE.

4 additional threats have been added to those of the **[PP_JC]**: T.APP_DATA_INTEGRITY, T.UNAUTH_CARD_MNGT, T.LIFE_CYCLE and T.UNAUTH_ACCESS. These threats have been added because card management becomes part of the TOE.

All OSPs from **[PP_JC]** are included in this ST, and 4 OSPs are added: OSPs OSP.SECURITY_DOMAINS, OSP.QUOTAS, OSP.KEY_GENERATION and
OSP.SHARE-CONTROL.OSP.SECURITY_DOMAINS and OSP.QUOTAS have been included because the security domain is part of the TOE.
All assumptions from **[PP_JC]** are included in this ST. However, some assumptions from the PP are removed in the ST for the following reasons: A.DELETION becomes irrelevant in this ST as card management, applet deletion included, is a TOE feature.
The assumption A.PRODUCTION has been added because of the TOE life cycle (the TOE can be delivered before step 6). This assumption doesn't mitigate any threat and doesn't fulfil any OSP meant to be addressed by security objectives for the TOE in the PP.

The statement of SPD is therefore consistent with those stated in **[PP_JC]**.

4.4.3 Security Objectives Consistency

The TOE objectives are a superset of those in **[PP_JC]**. Actually, all the TOE objectives from the PP are copied in the ST with the exception of the optional objective O.REMOTE, which is an objective for the not supported RMI functionality. Additionally, the objectives for the operational environment from the PP related the SCP (IC and Microkernel parts) are moved as TOE objectives in the ST. These objectives are: O.SCP.IC, O.SCP.RECOVERY, and O.SCP.SUPPORT.
The card manager becomes part of the TOE: the objective for the operational environment OE.CARD-MANAGEMENT is moved as TOE objective in the ST.
The optional security objective from the **[PP_JC]**, O.BIO-MNGT has been added because biometric template is part of the TOE

Objectives for the environment in this ST are identical to those in **[PP_JC]**. However, some objectives for the environment from the PP are removed in the ST for the following reasons:

- OE.CARD-MANAGEMENT, OE.SCP.IC, OE.SCP.RECOVERY, and OE.SCP.SUPPORT are transformed into TOE objectives. These objectives don't mitigate any threats of the PP and don't fulfil any OSPs meant to be addressed by security objectives for the TOE in the PP.

4.4.4 Consistency of the Security Objectives for the environment

The security objectives for the operational environment directly taken over from **[PP_JC]** are: OE.APPLET, OE.VERIFICATION and OE.CODE-EVIDENCE.
Others security objectives for the operational environment from **[PP_JC]** become objectives for the TOE.

Additionally, these security objectives for the operational environment are added to the ST: OE.SECURITY-DOMAINS, OE.QUOTAS, OE.SHARE-CONTROL, OE.KEY_GENERATION and OE.PRODUCTION.

4.4.5 Security Requirements Consistency

The set of SFRs is a superset of those in the **[PP_JC]** with the exception that the SFRs related to RMI functionality are not part of this security target. Actually, all the SFRs taken over from the PP are refined in the ST. Furthermore, the following SFRs, related to the IT requirements introduced in **[PP_JC]** by the Smart Group Platform and that are imposed on the operating system and the integrated circuit underlying the implementation of the Runtime Environment, have been added:

- FPT_RCV.3/OS;
- FPT_RCV.4/OS;
- FPT_FLS.1/OS;
- FPT_PHP.3/OS;

The following SFRs related to the Card Life Cycle Management have been added :

- FDP_ACC.1/CardLifeCycleManagement;
- FDP_ACF.1/CardLifeCycleManagement;
- FMT_MSA.1/CardLifeCycleManagement;
- FMT_MSA.3/CardLifeCycleManagement;
- FTP_ITC.1/CardLifeCycleManagement.

The following SFRs related to the PACE API have been added :

- FCS_CKM.2/PACE;
- FCS_CKM.3/PACE;
- FCS_COP.1/PACE.

5 SECURITY ASPECTS

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

5.1 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.

5.2 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.

5.3 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; (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; (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 those TSFs, is submitted to the same requirements. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

5.4 BYTECODE VERIFICATION

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

5.5 CARD MANAGEMENT

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

#.SID (1) Users and subjects of the TOE must be identified. (2) The identity of sensitive users and subjects associated with administrative and privileged roles must be particularly protected; this concerns the Java Card RE, the applets registered on the card, and especially the default applet and the currently selected applet (and all other active applets in Java Card System 2.2.x). A change of identity, especially standing for an administrative role (like an applet impersonating the Java Card RE), is a severe violation of the Security Functional Requirements (SFR). Selection controls the access to any data exchange between the TOE and the CAD and therefore, must be protected as well. The loading of a package or any exchange of data through the APDU buffer (which can be accessed by any applet) can lead to disclosure of keys, application code or data, and so on.

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

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

implementation-dependent. The only commitment is that deletion shall not jeopardize the TOE (or its assets) in case of failure (such as power shortage).
Deletion of a single applet instance and deletion of a whole package are functionally different operations and may obey different security rules. For instance, specific packages can be declared to be undeletable (for instance, the Java Card API packages), or the dependency between installed packages may forbid the deletion (like a package using super classes or super interfaces declared in another package).

5.6 SERVICES

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

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

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

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

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

#.PIN-MNGT The TOE shall provide a means to securely manage PIN objects. This includes: (1) Atomic update of PIN value and try counter, (2) No rollback on the PIN-checking function, (3) Keeping the PIN value (once initialized) secret (for instance, no clear-PIN-reading function), (4) Enhanced protection of PIN's security attributes (state, try counter…) in confidentiality and integrity.

#.SCP The smart card platform must be secure with respect to the SFRs. Then: (1) After a power loss, RF signal loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state. (2) It does not allow the SFRs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the Java Card API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System. (3) It provides secure low-level cryptographic processing to the Java Card System. (4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism. (5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low–level control accesses (segmentation fault detection). (6) It safely transmits low–level exceptions to the TOE (arithmetic exceptions, checksum errors), when applicable. Finally, it is required that (7) the IC is designed in accordance with a well-defined set of policies and standards (for instance, those specified in [PP0035]), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

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

6 Security Problem Definition

6.1 Assets

6.1.1 JavaCard System Protection Profile - Open Configuration

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.

6.1.1.1 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.

D.BIO

Biometric template. Only the fingerprint is in the scope of the TOE.

To be protected from unauthorized disclosure and modification.

6.1.1.2 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 JavaCard System.

To be protected from unauthorized disclosure and modification.

D.JCS_DATA

The internal runtime data areas necessary for the execution of the JavaCard VM, such as, for instance, the frame stack, the program counter, the class of an object, the length allocated for an array, any pointer used to chain data-structures.

To be protected from unauthorized disclosure or modification.

D.SEC_DATA

The runtime security data of the JavaCard 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.

6.1.2 Card Management

6.1.2.1 User Data

D.COMMAND

An APDU commands addressed to the Security Domains contains a request for a card management service. Valid requests come either from the Cardholder or from the Card Administrator. This asset shall be protected from unauthorized modification. Some specific card management commands, like those containing keys, shall be also protected from unauthorized disclosure.

6.1.2.2 TSF Data

D.GP_CODE

The code of the GlobalPlatform framework on the card. To be protected from unauthorized modification.

D.SD_KEYS

The cryptographic keys that the Security Domain uses for ensuring the integrity and origin of card management requests. This asset shall be protected from unauthorized disclosure and modification.

D.ISD_KEYS

During personalization, the cryptographic keys are stored in the Issuer Security Domain, the on-card representative of the Card Issuer. These keys needed to support several card management functions, like setting up a secure channel with the terminal. If the card is issued with Supplementary Security Domains, cryptographic keys of these Security Domain are also personalized. These assets shall be protected from disclosure and unauthorized modification.

6.2 Threats

This section describes the threats that concerned the TOE. Only the threat concerning *IDeal Citiz v2.1 open platform* platform are described, but threats on the table below (from [ST_IC]) must be considered:

T.Phys-Manipulation	Physical Manipulation
T.Phys-Probing	Physical Probing
T.Malfunction	Malfunction due to Environmental Stress
T.Leak-Inherent	Inherent Information Leakage
T.Leak-Forced	Forced Information Leakage
T.Abuse-Func	Abuse of Functionality
T.RND	Deficiency of Random Numbers
T.Mem-Access	Memory Access Violation

6.2.1 JavaCard System Protection Profile - Open Configuration

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

6.2.1.1 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, D.BIO and D.APP_KEYs.

T.CONFID-JCS-CODE

The attacker executes an application to disclose the JavaCard 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 JavaCard System. See #.CONFID-JCS-DATA for details.

Directly threatened asset(s): D.API_DATA, D.SEC_DATA, D.JCS_DATA and D.CRYPTO.

6.2.1.2 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, D.BIO and D.APP_KEYs.

T.INTEG-APPLI-DATA.LOAD

The attacker modifies (part of) the initialization data contained in an application package when the package is transmitted to the card for installation. See #.INTEG-APPLI-DATA for details.

Directly threatened asset(s): D.APP_I_DATA and D_APP_KEY.

T.INTEG-JCS-CODE

The attacker executes an application to alter (part of) the JavaCard 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) JavaCard 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.

6.2.1.3 IDENTITY USURPATION

T.SID.1

An applet impersonates another application, or even the JavaCard 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).

6.2.1.4 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.

6.2.1.5 DENIAL OF SERVICE

T.RESOURCES

An attacker prevents correct operation of the JavaCard System through consumption of some resources of the card: RAM or NVRAM. See #.RESOURCES for details.

Directly threatened asset(s): D.JCS_DATA.

6.2.1.6 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).

6.2.1.7 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.

6.2.2 Card Management

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 JavaCard 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.

T.APP_DATA_INTEGRITY

The attacker through a malicious applet loaded on the card modifies application data, application keys or authentication data. Directly threatened asset(s): **D.ISD_KEYS, D.API_DATA and D.APP_KEYs**.

T.UNAUTH_CARD_MNGT

The attacker performs unauthorized card management operations (for instance impersonates one of the actor represented on the card) in order to take benefit of the privileges or services granted to this actor on the card such as fraudulent:

- load of a package file;
- installation of a package file;
- extradition of a package file or an applet;
- personalization of an applet or a Security Domain;
- deletion of a package file or an applet;
- privileges update of an applet or a Security Domain.

Directly threatened asset(s): **D.ISD_KEYS, D.APP_KEYs, D.APP_C_DATA, D.APP_I_DATA and D.APP_CODE**.

T.LIFE_CYCLE

An attacker accesses to an application outside of its expected availability range thus violating irreversible life cycle phases of the application (for instance, an attacker re-personalizes the application). Directly threatened asset(s): **D.APP_I_DATA and D.APP_C_DATA**.

T.UNAUTH_ACCESS

By using the shareable object mechanism on which relies the communication between two applets, the attacker uses an applet on card to get access or to modify data from another applet that he should not have access to. Directly threatened asset(s): **all**.

6.3 Organisational Security Policies

This section describes the security policies of the TOE. The OSP from [ST_IC] must also be considered:

P.Process-TOE	Protection during TOE Development and Production
P.Add-Functions	Additional Specific Security Functionality

6.3.1 JavaCard 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 recommandations are applied in the application code.

6.3.2 TOE

OSP.SECURITY_DOMAINS

Security domains can be dynamically created, deleted and blocked during usage phase in post-issuance mode.

OSP.QUOTAS

Security domains are subject to quotas of memory at creation.

OSP.KEY_GENERATION

The personalizer must enforce a policy ensuring that generated keys cannot be accessed in plaintext.

Application Note:

This can be applied by encrypting the generated key just after its generation with the public key of the recipient.

OSP.SHARE-CONTROL

The Shareable interface functionality should be strictly controlled for all applications to prevent transitive data flows between applets (i.e., no resharing of a shareable object with a third applet) and thus prevent access to unauthorized data.

6.4 Assumptions

6.4.1 JavaCard System Protection Profile - Open Configuration

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

A.APPLET

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

6.4.2 TOE

A.PRODUCTION

Production and personalization environment if the TOE delivery occurs before step 6 of the TOE life cycle must be trusted and secure.

7 Security Objectives

7.1 Security Objectives for the TOE

This section describes the security objectives for the TOE. The security objectives described are those from the *IDeal Citiz v2.1 open platform* platform, but the security objectives for the TOE from [ST_IC], see table below, must be also considered.

Security Objectives	Description
O.Phys-Manipulation	Protection against Physical Manipulation
O.Phys-Probing	Protection against Physical Probing
O.Malfunction	Protection against Malfunction
O.Leak-Inherent	Protection against Inherent Information Leakage
O.Leak-Forced	Protection against Forced Information Leakage
O.Abuse-Func	Protection against Abuse of Functionality
O.Identification	TOE Identification
O.RND	Random Numbers
O.Add-Functions	Provide security functionality AES, 3DES, RSA, EC, SHA-2
O.Mem-Access	Capability to define restricted access memory areas

7.1.1 JavaCard System Protection Profile - Open Configuration

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

7.1.1.1 IDENTIFICATION

O.SID

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

7.1.1.2 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 JavaCard VM shall provide for an application to execute native code is the invocation of a method of the JavaCard 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 JavaCard VM does not disclose any information that was previously stored in that block.

O.RESOURCES

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

7.1.1.3 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.

O.BIO-MNGT

The TOE shall provide a means to securely manage biometric templates. This concerns the optional package javacardx.biometry of the Java Card platform.

O.TRANSACTION

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

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

7.1.1.4 OBJECT DELETION

O.OBJ-DELETION

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

7.1.1.5 APPLET MANAGEMENT

O.DELETION

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

O.LOAD

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

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

O.INSTALL

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

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

7.1.1.6 OPEN CONFIGURATION

O.SCP.IC

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

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

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

O.SCP.RECOVERY

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

This security objective refers to the security aspect #.SCP(1):

- The smart card platform must be secure with respect to the SFRs. Then after a power loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state.

O.SCP.SUPPORT

The SCP shall support the TSFs of the TOE.

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

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

(3) It provides secure low-level cryptographic processing to the JavaCard 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 JavaCard 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).

7.1.1.7 Card Management

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.

7.2 Security Objectives for the Operational Environment

7.2.1 JavaCard System Protection Profile - Open Configuration

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

OE.APPLET

No applet loaded post-issuance shall contain native methods.

OE.VERIFICATION

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

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 the used Protection Profile.

7.2.2 TOE

This section introduces the security objectives to be achieved by the environment associated to the TOE. The significant security objectives for the environment of the TOE are the ones linked to relevant assumptions and OSPs.

OE.SECURITY-DOMAINS

Security domains can be dynamically created, deleted and blocked during usage phase in post-issuance mode.

OE.QUOTAS

Security domains are subject to quotas of memory at creation.

OE.SHARE-CONTROL

All applications (basic and secure applications) must have means to identify the applications with whom they share data using the Shareable Interface.

OE.KEY_GENERATION

The personalizer must ensure that the generated keys cannot be accessed by unauthorized users.

OE.PRODUCTION

Production and personalization environment if the TOE delivery occurs before step 6 of the TOE life cycle must be trusted and secure. In particular, within the environments the corresponding guidance documents have to be taken into account.

7.3 Security Objectives Rationale

7.3.1 Threats

7.3.1.1 JavaCard System Protection Profile - Open Configuration

CONFIDENTIALITY

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

As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate 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, Biometry, PIN's are particular cases of an application's sensitive data (the JavaCard System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT, O.TRANSACTION). If the PIN class of the JavaCard API is used, the objective (O.FIREWALL) shall contribute in covering this threat by controlling the sharing of the global PIN between the applets.

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

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

T.CONFID-JCS-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the JavaCard 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 JavaCard 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 thisST by the objective for the environment OE.VERIFICATION.

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

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

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

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 JavaCard 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 JavaCard applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective O.NATIVE, so no application can run to modify a piece of code.

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

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

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

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

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

As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate 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, Biometry and PIN's are particular cases of an application's sensitive data (the JavaCard System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT, O.TRANSACTION). If the PIN class of the JavaCard API is used, the objective (O.FIREWALL) is also concerned.

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

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

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

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

T.INTEG-JCS-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the JavaCard 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 JavaCard applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective O.NATIVE, so no application can be run to modify a piece of code.

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

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

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

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

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

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.

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 JavaCard 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).

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

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

CARD MANAGEMENT

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

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

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

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

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.

7.3.1.2 Card Management

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

T.APP_DATA_INTEGRITY The security objective O.SCP.SUPPORT provides functionality to ensure atomicity of sensitive operations, secure low level access control and protection against bypassing of the security features of the TOE. In particular, it explicitly ensures the independent protection in integrity of the platform data. The security objectives covering the threat T.INTEG-APPLI-DATA also cover this threat.

T.UNAUTH_CARD_MNGT This threat is covered by the following security objectives:

- O.CARD-MANAGEMENT controls the access to card management functions such as the loading, installation, extradition or deletion of applets;

T.LIFE_CYCLE This threat is covered by the security objectives O.CARD-MANAGEMENT that controls the access to card management functions such as the loading, installation, extradition or deletion of applets and prevent attacks intended to modify or exploit the current life cycle of applications.

T.UNAUTH_ACCESS This threat is covered by the security objective on the operational environment of the TOE OE.SHARE-CONTROL which ensures that sharing objects functionality is strictly controlled to stop data transitive flows between applets and thus stop access to unauthorized data.

7.3.2 Organisational Security Policies

7.3.2.1 JavaCard System Protection Profile - Open Configuration

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.

7.3.2.2 TOE

OSP.SECURITY_DOMAINS This OSP is enforced by the security objective for the operational environment of the TOE OE.SECURITY-DOMAINS.

OSP.QUOTAS This OSP is enforced by the security objective for the operational environment of the TOE OE.QUOTAS.

OSP.KEY_GENERATION This OSP is directly enforced by the security objective for the operational environment of the TOE OE.KEY_GENERATION.

OSP.SHARE-CONTROL This OSP is directly enforced by the security objective for the operational environment of the TOE OE.SHARE-CONTROL.

7.3.3 Assumptions

7.3.3.1 JavaCard System Protection Profile - Open Configuration

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.

7.3.3.2 TOE

A.PRODUCTION This assumption is directly upheld by OE.PRODUCTION.

7.3.4 SPD and Security Objectives

Threats	Security Objectives	Rationale
T.CONFID-APPLI-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.BIO-MNGT, O.REALLOCATION, O.CARD-MANAGEMENT	Section 2.3.1
T.CONFID-JCS-CODE	OE.VERIFICATION, O.NATIVE, O.CARD-MANAGEMENT	Section 2.3.1
T.CONFID-JCS-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM, O.CARD-MANAGEMENT	Section 2.3.1
T.INTEG-APPLI-CODE	OE.VERIFICATION, O.NATIVE, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	Section 2.3.1
T.INTEG-APPLI-CODE.LOAD	O.LOAD, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	Section 2.3.1
T.INTEG-APPLI-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.GLOBAL_ARRAYS_INTEG, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.BIO-MNGT, O.REALLOCATION, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	Section 2.3.1
T.INTEG-APPLI-DATA.LOAD	O.LOAD, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	Section 2.3.1
T.INTEG-JCS-CODE	OE.VERIFICATION, O.NATIVE, O.CARD-MANAGEMENT, OE.CODE-EVIDENCE	Section 2.3.1
T.INTEG-JCS-DATA	O.SCP.RECOVERY, O.SCP.SUPPORT, OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM, O.CARD-MANAGEMENT	Section 2.3.1
T.SID.1	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.INSTALL, O.SID, O.CARD-MANAGEMENT	Section 2.3.1
T.SID.2	O.SCP.RECOVERY, O.SCP.SUPPORT, O.SID, O.OPERATE, O.FIREWALL, O.INSTALL	Section 2.3.1
T.EXE-CODE.1	OE.VERIFICATION, O.FIREWALL	Section 2.3.1
T.EXE-CODE.2	OE.VERIFICATION	Section 2.3.1
T.NATIVE	OE.VERIFICATION, OE.APPLET, O.NATIVE	Section 2.3.1

Threats	Security Objectives	Rationale
T.RESOURCES	O.INSTALL, O.OPERATE, O.RESOURCES, O.SCP.RECOVERY, O.SCP.SUPPORT	Section 2.3.1
T.DELETION	O.DELETION, O.CARD-MANAGEMENT	Section 2.3.1
T.INSTALL	O.INSTALL, O.LOAD, O.CARD-MANAGEMENT	Section 2.3.1
T.OBJ-DELETION	O.OBJ-DELETION	Section 2.3.1
T.PHYSICAL	O.SCP.IC	Section 2.3.1
T.APP_DATA_INTEGRITY	O.SCP.SUPPORT	Section 2.3.1
T.UNAUTH_CARD_MNGT	O.CARD-MANAGEMENT	Section 2.3.1
T.LIFE_CYCLE	O.CARD-MANAGEMENT	Section 2.3.1
T.UNAUTH_ACCESS	OE.SHARE-CONTROL	Section 2.3.1

Table 1- Threats and Security Objectives - Coverage

Security Objectives	Threats	Rationale
O.SID	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.1, T.SID.2	
O.FIREWALL	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.1, T.SID.2, T.EXE-CODE.1	
O.GLOBAL_ARRAYS_CONFID	T.CONFID-APPLI-DATA, T.SID.1	
O.GLOBAL_ARRAYS_INTEG	T.INTEG-APPLI-DATA, T.SID.1	
O.NATIVE	T.CONFID-JCS-CODE, T.INTEG-APPLI-CODE, T.INTEG-JCS-CODE, T.NATIVE	
O.OPERATE	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES	
O.REALLOCATION	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA	
O.RESOURCES	T.RESOURCES	
O.ALARM	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA	
O.CIPHER	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA	
O.KEY-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA	
O.PIN-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA	
O.BIO-MNGT	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA	
O.TRANSACTION	T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA	

Security Objectives	Threats	Rationale
O.OBJ-DELETION	T.OBJ-DELETION	
O.DELETION	T.DELETION	
O.LOAD	T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA.LOAD, T.INSTALL	
O.INSTALL	T.SID.1, T.SID.2, T.RESOURCES, T.INSTALL	
O.SCP.IC	T.PHYSICAL	
O.SCP.RECOVERY	T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES	
O.SCP.SUPPORT	T.APP_DATA_INTEGRITY, T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES	
O.CARD-MANAGEMENT	T.UNAUTH_CARD_MNGT, T.LIFE_CYCLE, T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA, T.INTEG-APPLI-DATA.LOAD, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.SID.1, T.DELETION, T.INSTALL	
OE.APPLET	T.NATIVE	
OE.VERIFICATION	T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-DATA, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.EXE-CODE.1, T.EXE-CODE.2, T.NATIVE	
OE.CODE-EVIDENCE	T.INTEG-APPLI-CODE, T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA, T.INTEG-APPLI-DATA.LOAD, T.INTEG-JCS-CODE	
OE.SECURITY-DOMAINS		
OE.QUOTAS		
OE.SHARE-CONTROL	T.UNAUTH_ACCESS	
OE.KEY_GENERATION		
OE.PRODUCTION		

Table 2 Security Objectives and Threats - Coverage

Organisational Security Policies	Security Objectives	Rationale
OSP.VERIFICATION	OE.VERIFICATION, OE.CODE-EVIDENCE	Section 2.3.2
OSP.SECURITY_DOMAINS	OE.SECURITY-DOMAINS	Section 2.3.2
OSP.QUOTAS	OE.QUOTAS	Section 2.3.2

Organisational Security Policies	Security Objectives	Rationale
OSP.KEY_GENERATION	OE.KEY_GENERATION	Section 2.3.2
OSP.SHARE-CONTROL	OE.SHARE-CONTROL	Section 2.3.2

Table 3 OSPs and Security Objectives - Coverage

Security Objectives	Organisational Security Policies	Rationale
O.SID		
O.FIREWALL		
O.GLOBAL_ARRAYS_CONFID		
O.GLOBAL_ARRAYS_INTEG		
O.NATIVE		
O.OPERATE		
O.REALLOCATION		
O.RESOURCES		
O.ALARM		
O.CIPHER		
O.KEY-MNGT		
O.PIN-MNGT		
O.BIO-MNGT		
O.TRANSACTION		
O.OBJ-DELETION		
O.DELETION		
O.LOAD		
O.INSTALL		
O.SCP.IC		
O.SCP.RECOVERY		
O.SCP.SUPPORT		
O.CARD-MANAGEMENT		
OE.APPLET		
OE.VERIFICATION	OSP.VERIFICATION	
OE.CODE-EVIDENCE	OSP.VERIFICATION	
OE.SECURITY-DOMAINS	OSP.SECURITY_DOMAINS	
OE.QUOTAS	OSP.QUOTAS	

Security Objectives	Organisational Security Policies	Rationale
OE.SHARE-CONTROL	OSP.SHARE-CONTROL	
OE.KEY_GENERATION	OSP.KEY_GENERATION	
OE.PRODUCTION		

Table 4 Security Objectives and OSPs - Coverage

Assumptions	Security Objectives for the Operational Environment	Rationale
A.APPLET	OE.APPLET	Section 2.3.3
A.VERIFICATION	OE.VERIFICATION, OE.CODE-EVIDENCE	Section 2.3.3
A.PRODUCTION	OE.PRODUCTION	Section 2.3.3

Table 5 Assumptions and Security Objectives for the Operational Environment - Coverage

Security Objectives for the Operational Environment	Assumptions	Rationale
OE.APPLET	A.APPLET	
OE.VERIFICATION	A.VERIFICATION	
OE.CODE-EVIDENCE	A.VERIFICATION	
OE.SECURITY-DOMAINS		
OE.QUOTAS		
OE.SHARE-CONTROL		
OE.KEY_GENERATION		
OE.PRODUCTION	A.PRODUCTION	

Table 6 Security Objectives for the Operational Environment and Assumptions - Coverage

8 Security Requirements

8.1 Security Functional Requirements

8.1.1 TOE

This section introduces the security functional requirements of the TOE that is composed of the open operating system *IDeal Citiz v2.1 open platform* and a chip M7892 B11. The following lists of SFRs are those from the security target [ST_IC] of the Infineon chip. They must be considered for the composite TOE, but they are not repeated in it. Details can be found on the security target [ST_IC].

- FRU_FLT.2 "Limited fault tolerance"
- FPT_FLS.1 "Failure with preservation of secure state"
- FMT_LIM.1 "Limited capabilities"
- FMT_LIM.2 "Limited availability"
- FAU_SAS.1 "Audit storage"
- FPT_PHP.3 "Resistance to physical attack"
- FDP_ITT.1 "Basic internal transfer protection"
- FPT_ITT.1 "Basic internal TSF data transfer protection"
- FDP_IFC.1 "Subset information flow control"
- FPT_TST.2 "Subset TOE security testing"
- FDP_ACC.1 "Subset access control"
- FDP_ACF.1 "Security attribute based access control"
- FMT_MSA.1 "Management of security attributes"
- FMT_MSA.3 "Static attribute initialisation"
- FMT_SMF.1 "Specification of Management functions"
- FCS_COP.1 "Cryptographic support"
- FCS_CKM.1 "Cryptographic key management"
- FDP_SDI.1 "Stored data integrity monitoring"
- FDP_SDI.2 "Stored data integrity monitoring and action"
- FCS_RNG.1 "Quality metric for random numbers"

The TOE does not provide the optional JCRMI functionality, therefore JCRMI related entities, subject, object, information, operation and security attribute, of the JavaCard Protection Profile [JCSPP] are excluded from the ST (the corresponding SFRs also).

8.1.2 JavaCard System Protection Profile - Open Configuration

This section states the security functional requirements for the JavaCard System - Open configuration. For readability and for compatibility with the original JavaCard System Protection Profile Collection - Standard 2.2 Configuration [PP/0305], requirements are arranged into groups. All the groups defined in the table below apply to this Protection Profile.

Group	Description

Group	Description
Core with Logical Channels (*CoreG_LC*)	The CoreG_LC contains the requirements concerning the runtime environment of the JavaCard System implementing logical channels. This includes the firewall policy and the requirements related to the JavaCard API. Logical channels are a JavaCard 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. JavaCard 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 JavaCard 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 JavaCard specification version 2.2 feature.
Secure carrier (*CarG*)	The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing, in those configurations that do not support on-card static or dynamic bytecode verification, the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.

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

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

Subject	Description
S.ADEL	The applet deletion manager which also acts on behalf of the card issuer. It may be an applet (**[JCRE]**, §11), but its role asks anyway for a specific treatment from the security viewpoint. This subject is unique and is involved in the ADEL security policy defined in §7.1.3.1.
S.APPLET	Any applet instance.
S.BCV	The bytecode verifier (BCV), which acts on behalf of the verification authority who is in charge of the bytecode verification of the packages. This subject is involved in the PACKAGE LOADING security policy defined in §7.1.7.

Subject	Description
S.CAD	The CAD represents the actor that requests, by issuing commands to the card, for RMI services. It also plays the role of the off-card entity that communicates with the S.INSTALLER.
S.INSTALLER	The installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the loading of packages and installation of applets.
S.JCRE	The runtime environment under which Java programs in a smart card are executed.
S.JCVM	The bytecode interpreter that enforces the firewall at runtime.
S.LOCAL	Operand stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.
S.MEMBER	Any object's field, static field or array position.
S.PACKAGE	A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of JavaCard technology, it defines either a user library, or one or several applets.
S.CM	Card Manager

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

Object	Description
O.APPLET	Any installed applet, its code and data.
O.CARD_LC	Card Manager Life Cycle State
O.CODE_PKG	The code of a package, including all linking information. On the JavaCard 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:

Security attribute	Description/Value

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.
Applet Privileges	Privileges of an applet
CARD_LC State	Life Cycle States: OP_READY, INITIALIZED, SECURED, CARD_LOCKED, and TERMINATED
Class	Identifies the implementation class of the remote object.
Context	Package AID or "JavaCard RE".
Currently Active Context	Package AID or "JavaCard RE".
Dependent package AID	Allows the retrieval of the Package AID and Applet's version number (**[JCVM]**, §4.5.2).
ExportedInfo	Boolean (indicates whether the remote object is exportable or not).
Identifier	The Identifier of a remote object or method is a number that uniquely identifies the remote object or method, respectively.
LC Selection Status	Multiselectable, Non-multiselectable or "None".
LifeTime	CLEAR_ON_DESELECT or PERSISTENT (*).
Owner	The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the package). The owner of a remote object is the applet instance that created the object.
Package AID	The AID of each package indicated in the export file.
Registered Applets	The set of AID of the applet instances registered on the card.
Resident Packages	The set of AIDs of the packages already loaded on the card.
Selected Applet Context	Package AID or "None".
Sharing	Standards, SIO, JavaCard RE entry point or global array.

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

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

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

Operation	Description
OP.ARRAY_ACCESS(O.JAVAOBJECT, field)	Read/Write an array component.
OP.CREATE(Sharing, LifeTime) (*)	Creation of an object (new or makeTransient call).
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed applet and its objects, either logically or physically.
OP.DELETE_PCKG(O.CODE_PKG,...)	Delete a package, either logically or physically.
OP.DELETE_PCKG_APPLET(O.CODE_PKG,...)	Delete a package and its installed applets, either logically or physically.
OP.INSTANCE_FIELD(O.JAVAOBJECT, field)	Read/Write a field of an instance of a class in the Java programming language.
OP.INVK_VIRTUAL(O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object).
OP.INVK_INTERFACE(O.JAVAOBJECT, method, arg1,...)	Invoke an interface method.
OP.JAVA(...)	Any access in the sense of **[JCRE]**, §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.SET_CARD_STATE(…)(**)	Set Card Life Cycle State
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see **[JCRE]**, §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.

(**) This operation, not present in [PP JC] has been added for the Card Life Cycle Management SFRs

8.1.2.1 CoreG_LC Security Functional Requirements

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

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.

FDP_ACF.1/FIREWALL Security attribute based access control

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

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

.

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

- **R.JAVA.1 ([JCRE], §6.2.8): S.PACKAGE may freely perform OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW or**

OP.TYPE_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".

- **R.JAVA.2 ([JCRE], §6.2.8): S.PACKAGE may freely perform OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE or OP.THROW upon any O.JAVAOBJECT whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if O.JAVAOBJECT's Context attribute has the same value as the active context.**
- **R.JAVA.3 ([JCRE], §6.2.8.10): S.PACKAGE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT whose Sharing attribute has value "SIO" only if O.JAVAOBJECT is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.**
- **R.JAVA.4 ([JCRE], §6.2.8.6): S.PACKAGE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if the invoked interface method extends the Shareable interface and one of the following conditions applies:**
  - **a) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable",**
  - **b) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Non-multiselectable", and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute Active Applets.**
- **R.JAVA.5: S.PACKAGE may perform OP.CREATE only if the value of the Sharing parameter is "Standard"**.

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

- **1) The subject S.JCRE can freely perform OP.JAVA(") and OP.CREATE, with the exception given in FDP_ACF.1.4/FIREWALL, provided it is the Currently Active Context.**
- **2) The only means that the subject S.JCVM shall provide for an application to execute native code is the invocation of a JavaCard 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**.

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)**.

FDP_IFF.1/JCVM Simple security attributes

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

Subjects	Security attributes
S.JCVM	Currently Active Context

.

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

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

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

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

FDP_IFF.1.5/JCVM The TSF shall explicitly deny an information flow based on the following rules: **one of the conditions in the element FDP_IFF.1.2-JCVM above is not satisfied**.

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 **Selected Applet Context** to **the JavaCard RE**.

Remark The TOE is compliant with GlobalPlatform's specifications and includes the GlobalPlatform Environment that provide an API to applications, command dispatch, Card Content management and manages the installation of applications to the card and loading of these latter. These functions are available if not provided by the JavaCard RE, or if provided but in a way not complying with GlobalPlatform's specifications.

FMT_MSA.1/JCVM Management of security attributes

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

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 initialisation

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

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

FMT_MSA.3/JCVM Static attribute initialisation

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

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

FMT_SMF.1 Specification of Management Functions

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

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

FMT_SMR.1 Security roles

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

- **JavaCard RE (JCRE),**
- **JavaCard VM (JCVM)**.

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

Application Programming Interface

The following SFRs are related to the JavaCard API.

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

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

FCS_CKM.1 Cryptographic key generation

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

Iteration	Algorithm	Key Size	Standard
RSA-CRT	RSA-CRT key generation	768 to 3072 bits	[FIPS_186-4]
EC_FP	Elliptic Curve Prime Field Algorithm key generation for ECDSA & ECDH	112, 128, 160, 192, 224, 256, 384, 512, 521 bits	[FIPS_186-4]

.

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 **Diffie-Hellman key agreement based on elliptic curve cryptography algorithm defined in [JCAPI]:**

- **ALG_EC_SVDP_DH**
- **ALG_EC_SVDP_DH_KDF**
- **ALG_EC_SVDP_DH_PLAIN**
- **ALG_EC_SVDP_DHC**
- **ALG_EC_SVDP_DHC_KDF**
- **ALG_EC_SVDP_DHC_PLAIN**

that meets the following: **[NIST_SP800-56A] and [IEEE1363]**.

FCS_CKM.3 Cryptographic key access

FCS_CKM.3.1 The TSF shall perform **see table below** in accordance with a specified cryptographic key access method **see table below** that meets the following: **see table below**

Iteration	Key Access	Method	Standard
JCS	key access	using API of class Key	[JCAPI]

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 **overwritting of data** that meets the following: **none**.

FCS_COP.1 Cryptographic operation

FCS_COP.1 The TSF shall perform **see table below** in accordance with a specified cryptographic algorithm **see table below** and cryptographic key sizes **see table below** that meet the following: **see tables below**

Asymetric encryption/decryption and signature/verification:

Iteration	Operation	Algorithm	Key Size	Standard
RSA-CRT_SIG	user data signature generation and verification	RSA CRT	768 to 3072 bits	[PKCS#1_v2.1]
RSA-CRT_CPH	user data encryption and decryption	RSA CRT	768 to 3072 bits	[PKCS#1_v2.1]
ECDH	key agreement	ECDH	112, 128, 160, 192, 224, 256, 384, 512, 521	[FIPS_186-4], [IEEEP1363]
ECDSA	user data signature generation and verification	Elliptic Curve Digital Signature Algorithm	112, 128, 160, 192, 224, 256, 384, 512, 521	[FIPS_186-4], [IEEEP1363]

Symetric encryption/decryption:

Iteration	Operation	Algorithm	Key Size	Standard
TDES	encryption and decryption of application instance's data	Triple DES ECB or CBC mode	128, 192 bits	[ANSI_X9.52], [FIPS_46-3]
AES	encryption and decryption of application instance's data	AES ECB or CBC mode	128, 192, 256 bits	[FIPS_197]

Hash:

Iteration	Operation	Algorithm	Hash Size	Standard
SHA	user data hash generation	Secure Hash Algorithm	160, 224, 256, 384, 512 bits	[FIPS_180-4]

MAC:

Iteration	Operation	Algorithm	Key Size	Standard
HMAC	computation of message authentication code based on hash algorithm	Keyed-Hash Message Authentication Code	160, 224, 256, 384, 512 bits	[FIPS_198-1]
CMAC	computation of message authentication code based on cipher algorithm	Cipher-based Message Authentication Code	128, 192, 256 bits	[NIST_SP800-38B]

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 **object O.JAVAOBJECT**.

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

Card Security Management

FAU_ARP.1 Security alarms

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

- **throw an exception,**
- **lock the card session,**
- **reinitialize the JavaCard System and its data**
- **Increment an application error counter and mute the card. When the application error counter reaches its maximum value, the application is locked.**
- **Increment a card error counter and mute the card. When the card error counter reaches its maximum value, the card is terminated.**

upon detection of a potential security violation.

Refinement:

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

- CAP file inconsistency,
- typing error in the operands of a bytecode,
- applet life cycle inconsistency,
- card tearing (unexpected removal of the Card out of the CAD) and power failure,
- abort of a transaction in an unexpected context, (see abortTransaction(), [JCAPI] and (**[JCRE]**, §7.6.2)
- violation of the Firewall or JCVM SFPs,
- unavailability of resources,
- array overflow,
- hidden alarm signalled by the application layer,
- failure of explicit integrity checks triggered by the application layer.

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:

- **User Packages**

- **PIN Objects**
- **OwnerBioTemplate Objects created by javacardx.biometry.buildBioTemplate()**
- **Cipher Objects created by the javacardx.crypto.Cipher.getInstance() method**
- **Signature Objects created by the javacard.security.Signature.getInstance() method**
- **RandomData Objects created by the javacard.security.RandomData.getInstance() method**
- **MessageDigest Objects created by the javacard.security.MessageDigest.getInstance() method**
- **InitializedMessageDigest Objects created by the javacard.security.MessageDigest.getInitializedMessageDigestInstance() method**
- **Key Objects created by the javacard.security.KeyBuilder.buildKey() method**.

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

FPR_UNO.1 Unobservability

FPR_UNO.1.1 The TSF shall ensure that **any off-card or on-card subject other than the runtime environment (S.JCRE) and the currently running applet** are unable to observe the operation **PIN verification operations, bio datamatching operation, encryption and decryption operations, signature generation and verification operations, random data generation operations, key agreement operations, key access operations, signalling of a hidden alarm** on **PIN Objects, Cipher objects, signature objects, random data objects, key pair objects, key agreement objects, key objects, or the currently running applet** by **the subject S.JCRE or the currently running applet**.

FPT_FLS.1 Failure with preservation of secure state

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

FPT_TDC.1 Inter-TSF basic TSF data consistency

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

FPT_TDC.1.2 The TSF shall use

- **the rules defined in [JCVM] specification,**
- **the API tokens defined in the export files of reference implementation**

when interpreting the TSF data from another trusted IT product.

AID Management

FIA_ATD.1/AID User attribute definition

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

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

Refinement:

"Individual users" stand for applets.

FIA_UID.2/AID User identification before any action

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

FIA_USB.1/AID User-subject binding

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

FIA_USB.1.2/AID The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **the AID of the package acting on the behalf of the user shall be equal to the Package AID security attribute of the user**.

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: **the Package AID security attribute of a user shall not be modified**.

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 registered applets' AIDs**.

8.1.2.2 InstG Security Functional Requirements

This group consists of the SFRs related to the installation of the applets, which addresses security aspects outside the runtime. The installation of applets is a critical phase, which lies partially out of the boundaries of the firewall, and therefore requires specific treatment. In this ST, loading a package or installing an applet modeled as importation of

user data (that is, user application's data) with its security attributes (such as the parameters of the applet used in the firewall rules).

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

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

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

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

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

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

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

FMT_SMR.1/Installer Security roles

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

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

FPT_FLS.1/Installer Failure with preservation of secure state

FPT_FLS.1.1/Installer The TSF shall preserve a secure state when the following types of failures occur: **the installer fails to load/install a package/applet as described in [JCRE] §11.1.4**.

FPT_RCV.3/Installer Automated recovery without undue loss

FPT_RCV.3.1/Installer When automated recovery from **any package loading session interruption or failure, any applet installation interruption or failure** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2/Installer For **any package loading session interruption or failure, any applet installation interruption or failure**, 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 **the loss of the package or applet installed** 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.

8.1.2.3 ADELG Security Functional Requirements

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

FDP_ACC.2/ADEL Complete access control

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

Refinement:

The operations involved in the policy are:

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

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

FDP_ACF.1/ADEL Security attribute based access control

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

Subject/Object	Attributes
S.JCVM	Active Applets
S.JCRE	Selected Applet Context, Registered Applets, Resident Packages

Subject/Object	Attributes
O.CODE_PKG	Package AID, Dependent Package AID, Static References
O.APPLET	Applet Selection Status
O.JAVAOBJECT	Owner, Remote

.

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

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

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

owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted

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

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

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**.

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 JavaCard RE**.

FMT_MSA.3/ADEL Static attribute initialisation

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

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

FMT_SMF.1/ADEL Specification of Management Functions

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

FMT_SMR.1/ADEL Security roles

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

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

FPT_FLS.1/ADEL Failure with preservation of secure state

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

8.1.2.4 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**.

8.1.2.5 CarG Security Functional Requirements

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

FCO_NRO.2/CM Enforced proof of origin

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

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

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

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.

FDP_IFF.1/CM Simple security attributes

FDP_IFF.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** based on the following types of subject and information security attributes: **Subjects (for all secure channel protocol implementations): S.CAD and S.BCV, S.CM, Subjects (package loading): S.CM's Security Domain with Authorized Management (i.e. Card Issuer Security domain) or Delegated Management privilege, S.CM's Security Domain with Mandated DAP verification privilege (i.e. Verification Authority Security Domain), S.CM's Security Domain with Token verification privilege, S.CM's OPEN. Information: I.APDU INSTALL [for load] command data, I.APDU LOAD command data. Security attributes: Secure channel key(s), Security level, Privileges.**

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 rules describing the communication protocol (SCP02 or SCP03) used by the CAD and the card for transmitting a new package**.

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

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

FDP_IFF.1.5/CM The TSF shall explicitly deny an information flow based on the following rules: **when at least one of the conditions listed in the element FDP_IFF.1.2 does not hold**.

FDP_UIT.1/CM Data exchange integrity

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

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

FIA_UID.1/CM Timing of identification

FIA_UID.1.1/CM The TSF shall allow **application selection, initializing a secure channel with the card and requesting data that identifies the card or the Card Issuer** 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 **delete, modify, query and change_default** the security attributes **key value, key version, key identifier** to **Security Domain**.

FMT_MSA.3/CM Static attribute initialisation

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

FMT_MSA.3.2/CM The TSF shall allow the **currently selected Security Domain** 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:

- **loading**
- **installation**
- **extradition**
- **content removal**
- **application personalization**
- **card life cycle**

Application Note:

These Management functions are specified in GlobalPlatform specifications [GP]

FMT_SMR.1/CM Security roles

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

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

Application Note:

A Security Domain is just the on-card counterpart of a representative of the Card Issuer. It is introduced as a separate role in order to distinguish an application acting inside the smart card on behalf of the Card Issuer from the Card Administrator.

FTP_ITC.1/CM Inter-TSF trusted channel

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

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

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

8.1.3 Operating System

The Smart Card Platform group introduced in [PP_JC] specifies the IT requirements that are imposed on the Operating System and the Integrated Circuit underlying the implementation of the Runtime Environment. Because of the modification in the scope of evaluation, which does include in this Security Target the Operating System and the Integrated Circuit, those requirements on the IT environment become requirements on the TOE itself.

8.1.3.1 OSG Security Functionnal Requirements

FPT_RCV.3/OS Automated recovery without undue loss

FPT_RCV.3.1/OS When automated recovery from **security policy violation** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2/OS For **execution access to a memory zone reserved for TSF data, writing access to a memory zone reserved for TSF's code, and any segmentation fault performed by a JavaCard applet**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/OS The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding

- **the contents of JavaCard static fields, instance fields, and array positions that fall under the scope of an open transaction;**

- **the JavaCard objects that were allocated into the scope of an open transaction;**
- **the contents of JavaCard transient objects;**
- **any possible Executable Load File being loaded when the failure occurred**

for loss of TSF data or objects under the control of the TSF.

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

FPT_RCV.4/OS Function recovery

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

FPT_FLS.1/OS Failure with preservation of secure state

FPT_FLS.1.1/OS The TSF shall preserve a secure state when the following types of failures occur:

- **invalid reference exception;**
- **code or data integrity failure;**
- **power loss;**
- **NVM failure**.

FPT_PHP.3/OS Resistance to physical attack

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

8.1.4 Card Life Cycle Management SFRs

These SFRs have been added to cover the O.CARD-MANAGEMENT objective as it becomes a TOE objective

FDP_ACC.1/CardLifeCycleManagement Subset Access Control

FDP_ACC.1.1/CardLifeCycleManagement The TSF shall enforce the **CARD LIFE CYCLE MANAGEMENT access control SFP** on :

Subjects: S.CM, S.APPLET, S.CAD;

Operations: OP.SET_CARD_STATE.

Objects: O.CARD_LC.

FDP_ACF.1/CardLifeCycleManagement Security Attribute based Access Control

FDP_ACF.1.1/CardLifeCycleManagement The TSF shall enforce the **CARD LIFE CYCLE MANAGEMENT access control SFP** to objects based on :

the security attributes of O.CARD_LC: State as defined in [GP_CS] Section 5.1: OP_READY, INITIALIZED, SECURED, CARD_LOCKED, TERMINATED.

the security attributes of S.APPLET: Privileges as defined in [GP_CS] Section 6.6 : Card Lock, Card Terminate

FDP_ACF.1.2/CardLifeCycleManagement The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:
S.CM is allowed to set the Card Life Cycle to INITIALIZED, SECURED, CARD_LOCKED, and TERMINATED as defined in [GP_CS] Section 5.1.2.
S.APPLET is allowed to set the Card Life Cycle to CARD_LOCKED if S.APPLET has the Privilege Card Lock
S.APPLET is allowed to set the Card Life Cycle to TERMINATED if S.APPLET has the Privilege Card Terminate.

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

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

If the O.CARD_LC State=TERMINATED, the TOE is disabled, and the access of subjects is no more allowed.

Or when at least one of the rules defined by [GP_CS] does not hold.

FMT_MSA.1/CardLifeCycleManagement Management of Security Attributes

FMT_MSA.1.1/CardLifeCycleManagement The TSF shall enforce the **CARD LIFE CYCLE MANAGEMENT access control SFP** to restrict the ability to **modify** the security attributes **O.CARD_LC State** to **S.CM and S.APPLET.**

Application Note: S.APPLET should be an application with Card Lock, Card Terminate Privileges as defined in [GP_CS]

FMT_MSA.3/CardLifeCycleManagement Static Attribute Initialization

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

FMT_MSA.3.2/CardLifeCycleManagement 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**.**

FTP_ITC.1/CardLifeCycleManagement Inter-TSF Trusted Channel

FTP_ITC.1.1/CardLifeCycleManagement 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/CardLifeCycleManagement The TSF shall permit **the CAD placed in the card issuer secured environment** to initiate communication via the trusted channel.

FTP_ITC.1.3/CardLifeCycleManagement TSF shall initiate communication via the trusted channel for **setting the Card Life Cycle State**.

8.1.5 SFRs for PACE API

These SFRs have been added to cover the PACE API of the TOE.

This group of SFRs apply only if the TOE provides PACE API, as it is removable.

FCS_CKM.2/PACE Cryptographic key distribution

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **Diffie-Hellman key agreement based on the ephemeral domain parameters and generates the session keys for encryption/decryption and authentication** that meets the following: **none**

FCS_CKM.3/PACE Cryptographic key access

FCS_CKM.3.1 The TSF shall perform **see table below** in accordance with a specified cryptographic key access method **see table below** that meets the following: **see table below**

Iteration	Key Access	Method	Standard
ENC	Key for encryption/decryption	getSessionKeyEnc	none

MAC	Key for authentication	getSessionKeyMac	none

FCS_COP.1/PACE Cryptographic operation

FCS_COP.1.1 The TSF shall perform **see table below** in accordance with a specified cryptographic algorithm **see table below** and cryptographic key sizes **see table below** that meet the following: **see tables below**

Operation	Algorithm	Key Size	Standard
Generates and encrypts the nonce, which is used to perform the following operations of a SAC establishment	3DES and AES, in CBC mode	112 bits for TDES and 128, 192 or 256 bits for AES	[ICAO-SAC]
Computes the ephemeral domain parameters of the SAC establishment	3DES and AES, in CBC mode	112 bits for TDES and 128, 192 or 256 bits for AES	[ICAO-SAC]
Performs a mutual authentication : exchange and verify an authentication token	Retail MAC with 3DES, AES	112 bits for TDES and 128, 192 or 256 bits for AES	[ICAO-SAC]

8.2 Security Assurance Requirements

The Evaluation Assurance Level is EAL5 augmented with AVA_VAN.5 and ALC_DVS.2.

8.3 Security Requirements Rationale

8.3.1 Objectives

8.3.1.1 Security Objectives for the TOE

JavaCard System Protection Profile - Open Configuration

IDENTIFICATION

O.SID Subjects' identity is AID-based (applets, packages), and is met by the following SFRs: FDP_ITC.2/Installer, FIA_ATD.1/AID, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_MSA.1/ADEL, FMT_MSA.1/CM, FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.3/CM, FMT_SMF.1/CM, FMT_SMF.1/ADEL, FMT_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).

EXECUTION

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

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

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

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

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

O.OPERATE The TOE is protected in various ways against applets' actions (FPT_TDC.1), the FIREWALL access control policy FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL, and is able to detect and block various failures or security violations during usual working (FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FAU_ARP.1). Its security-critical parts and procedures are also protected: safe recovery from failure is ensured (FPT_RCV.3/Installer), applets' installation may be cleanly aborted (FDP_ROL.1/FIREWALL), communication with external users and their internal subjects is well-controlled (FDP_ITC.2/Installer, FIA_ATD.1/AID, FIA_USB.1/AID) to prevent alteration of TSF data (also protected by components of the FPT class).

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

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

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

SERVICES

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

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

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

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.BIO-MNGT This 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, FPR_UNO.1, FDP_ROL.1/FIREWALL and FDP_SDI.2 security functional requirements. The applets that manage biometric templates rely on the security functions that implement these SFRs. The firewall security functions (FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL) shall protect the access to private and internal data of the templates. Note that the objective applies only to configurations including the javacardx.biometry package defined in [JCAPI].

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).

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.

APPLET MANAGEMENT

O.DELETION This security objective specifies that applet and package deletion must be secure. The non-introduction of security holes is ensured by the ADEL access control policy (FDP_ACC.2/ADEL, FDP_ACF.1/ADEL). The integrity and confidentiality of data that does not belong to the deleted applet or package is a by-product of this policy as well. Non-accessibility of deleted data is met by FDP_RIP.1/ADEL and the TSFs are protected against possible failures of the deletion procedures (FPT_FLS.1/ADEL, FPT_RCV.3/Installer). The security functional requirements of the class FMT (FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL) included in the group ADELG also contribute to meet this objective.

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

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

OPEN CONFIGURATION

O.SCP.IC This security objective is ensure by IC related requirements FPT_PHP.3/OS and FPT_FLS.1/OS which refer to the general security of the underlying security IC to resist physical manipulation and probing and reacting to induced failures. Furthermore, the protection of key material as mandated by the objective is additionally enforced within the implementation of the JavaCard platform SFR FCS_COP.1 that offers cryptographic services to the application layer.

O.SCP.RECOVERY This security objective is ensure by FPT_RCV.3/OS and FPT_RCV.4/OS

O.SCP.SUPPORT This security objective is ensure by several SFRs:

- The general tamper resistance of the underlying security IC (FPT_PHP.3/OS) contributes to the protection of JavaCard system code and data. Furtermore, sensitive data objects are additionally protected by the integrity protection mechanisms provided by FDP_SDI.2. The enforcement of the domain separation mechanisms is part of the protection of the implementation of FMT_MSA.2/FIREWALL_JCVM and FMT_MSA.3/FIREWALL.
- The basic cryptographic support provided by the low-level parts of the system is considered within the implementation of FCS_COP.1. The same services are also used by other security implementation parts of the system like the secure channel protocol implementation in the Card Management component.
- The objective to ensure the atomicity of updates on persistent data is covered by the additonal requirements FPT_RCV.3/OS, FPT_RCV.4/OS.
- The security functional requirement FPT_FLS.1/OS which is related to the basic memory protection mechanisms of the underlying IC establishes a first level of protection. Additionally, the low-level basic operating system implements a structured memory management offered as a service to the rest of the system.

CARD MANAGEMENT

O.CARD-MANAGEMENT This security objective shall control the access to the card and implement the card issuers policy and is met by the components FDP_ACC.1/CardLifeCycleManagement,

FDP_ACF.1/CardLifeCycleManagement, FMT_MSA.1/CardLifeCycleManagement, FMT_MSA.3/CardLifeCycleManagement, and FTP_ITC.1/CardLifeCycleManagement.

8.3.2 Rationale tables of Security Objectives and SFRs

Security Objectives	Security Functional Requirements	Rationale
O.SID	FIA_ATD.1/AID, FIA_UID.2/AID, FMT_MSA.1/JCRE, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_MSA.3/FIREWALL, FMT_MSA.1/CM, FMT_MSA.3/CM, FDP_ITC.2/Installer, FMT_SMF.1/CM, FMT_SMF.1/ADEL, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FIA_USB.1/AID, FMT_MSA.1/JCVM, FMT_MSA.3/JCVM	Section 3.3.1
O.FIREWALL	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FMT_SMR.1/Installer, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_MSA.3/FIREWALL, FMT_SMR.1, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FMT_MSA.1/JCRE, FDP_ITC.2/Installer, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_SMF.1, FMT_MSA.2/FIREWALL_JCVM, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_MSA.1/JCVM, FMT_MSA.3/JCVM	Section 3.3.1
O.GLOBAL_ARRAYS_CONFID	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FDP_RIP.1/bArray, FDP_RIP.1/APDU, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT	Section 3.3.1
O.GLOBAL_ARRAYS_INTEG	FDP_IFC.1/JCVM, FDP_IFF.1/JCVM	Section 3.3.1
O.NATIVE	FDP_ACF.1/FIREWALL	Section 3.3.1
O.OPERATE	FAU_ARP.1, FDP_ROL.1/FIREWALL, FIA_ATD.1/AID, FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FDP_ITC.2/Installer, FPT_RCV.3/Installer, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FPT_TDC.1, FIA_USB.1/AID	Section 3.3.1
O.REALLOCATION	FDP_RIP.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS	Section 3.3.1

Security Objectives	Security Functional Requirements	Rationale
O.RESOURCES	FAU_ARP.1, FDP_ROL.1/FIREWALL, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMR.1/ADEL, FPT_FLS.1/Installer, FPT_FLS.1/ODEL, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_RCV.3/Installer, FMT_SMR.1/CM, FMT_SMF.1/ADEL, FMT_SMF.1/CM, FMT_SMF.1, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE	Section 3.3.1
O.ALARM	FPT_FLS.1/Installer, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL, FAU_ARP.1	Section 3.3.1
O.CIPHER	FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FCS_COP.1, FPR_UNO.1, FCS_CKM.2/PACE, FCS_CKM.3/PACE, FCS_COP.1/PACE	Section 3.3.1
O.KEY-MNGT	FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FCS_COP.1, FPR_UNO.1, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_SDI.2, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FCS_CKM.2/PACE, FCS_CKM.3/PACE, FCS_COP.1/PACE	Section 3.3.1
O.PIN-MNGT	FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FPR_UNO.1, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FDP_ROL.1/FIREWALL, FDP_SDI.2, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL	Section 3.3.1
O.BIO-MNGT	FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FPR_UNO.1, FDP_ROL.1/FIREWALL, FDP_SDI.2, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL	Section 3.3.1
O.TRANSACTION	FDP_ROL.1/FIREWALL, FDP_RIP.1/ABORT, FDP_RIP.1/ODEL, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FDP_RIP.1/OBJECTS	Section 3.3.1
O.OBJ-DELETION	FDP_RIP.1/ODEL, FPT_FLS.1/ODEL	Section 3.3.1
O.DELETION	FDP_ACC.2/ADEL, FDP_ACF.1/ADEL, FDP_RIP.1/ADEL, FPT_FLS.1/ADEL, FPT_RCV.3/Installer, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL	Section 3.3.1

Security Objectives	Security Functional Requirements	Rationale
O.LOAD	FCO_NRO.2/CM, FDP_IFC.2/CM, FDP_IFF.1/CM, FDP_UIT.1/CM, FIA_UID.1/CM, FTP_ITC.1/CM	Section 3.3.1
O.INSTALL	FDP_ITC.2/Installer, FPT_RCV.3/Installer, FPT_FLS.1/Installer	Section 3.3.1
O.SCP.IC	FPT_FLS.1/OS, FCS_COP.1, FPT_PHP.3/OS	Section 3.3.1
O.SCP.RECOVERY	FPT_RCV.4/OS, FPT_RCV.3/OS	Section 3.3.1
O.SCP.SUPPORT	FPT_FLS.1/OS, FPT_RCV.4/OS, FPT_RCV.3/OS, FPT_PHP.3/OS, FDP_SDI.2, FMT_MSA.3/FIREWALL, FMT_MSA.2/FIREWALL_JCVM, FCS_COP.1	Section 3.3.1
O.CARD-MANAGEMENT	FDP_ACC.1/CardLifeCycleManagement, FDP_ACF.1/CardLifeCycleManagement, FMT_MSA.1/CardLifeCycleManagement, FMT_MSA.3/CardLifeCycleManagement, FTP_ITC.1/CardLifeCycleManagement.	Section 3.3.1

Table 7 Security Objectives and SFRs - Coverage

Security Functional Requirements	Security Objectives	Rationale
FDP_ACC.2/FIREWALL	O.FIREWALL, O.OPERATE, O.PIN-MNGT, O.BIO-MNGT	
FDP_ACF.1/FIREWALL	O.FIREWALL, O.NATIVE, O.OPERATE, O.PIN-MNGT, O.BIO-MNGT	
FDP_IFC.1/JCVM	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG	
FDP_IFF.1/JCVM	O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG	
FDP_RIP.1/OBJECTS	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT, O.TRANSACTION	
FMT_MSA.1/JCRE	O.SID, O.FIREWALL	
FMT_MSA.1/JCVM	O.SID, O.FIREWALL	
FMT_MSA.2/FIREWALL_JCVM	O.FIREWALL, O.SCP.SUPPORT	

Security Functional Requirements	Security Objectives	Rationale
FMT_MSA.3/FIREWALL	O.SID, O.FIREWALL, O.SCP.SUPPORT	
FMT_MSA.3/JCVM	O.SID, O.FIREWALL	
FMT_SMF.1	O.FIREWALL, O.RESOURCES	
FMT_SMR.1	O.FIREWALL, O.RESOURCES	
FCS_CKM.1	O.CIPHER, O.KEY-MNGT	
FCS_CKM.2	O.CIPHER, O.KEY-MNGT	
FCS_CKM.3	O.CIPHER, O.KEY-MNGT	
FCS_CKM.4	O.CIPHER, O.KEY-MNGT	
FCS_COP.1	O.CIPHER, O.KEY-MNGT, O.SCP.IC, O.SCP.SUPPORT	
FDP_RIP.1/ABORT	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT, O.TRANSACTION	
FDP_RIP.1/APDU	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT,O.TRANSACTION	
FDP_RIP.1/bArray	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT,O.TRANSACTION	
FDP_RIP.1/KEYS	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT,O.TRANSACTION	
FDP_RIP.1/TRANSIENT	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION	
FDP_ROL.1/FIREWALL	O.OPERATE, O.RESOURCES, O.PIN-MNGT, O.BIO-MNGT, O.TRANSACTION	
FAU_ARP.1	O.OPERATE, O.RESOURCES, O.ALARM	
FDP_SDI.2	O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT, O.SCP.SUPPORT	

Security Functional Requirements	Security Objectives	Rationale
FPR_UNO.1	O.CIPHER, O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT	
FPT_FLS.1	O.OPERATE, O.RESOURCES, O.ALARM	
FPT_TDC.1	O.OPERATE	
FIA_ATD.1/AID	O.SID, O.OPERATE	
FIA_UID.2/AID	O.SID	
FIA_USB.1/AID	O.SID, O.OPERATE	
FMT_MTD.1/JCRE	O.SID, O.FIREWALL, O.RESOURCES	
FMT_MTD.3/JCRE	O.SID, O.FIREWALL, O.RESOURCES	
FDP_ITC.2/Installer	O.SID, O.FIREWALL, O.OPERATE, O.INSTALL	
FMT_SMR.1/Installer	O.FIREWALL, O.RESOURCES	
FPT_FLS.1/Installer	O.OPERATE, O.RESOURCES, O.ALARM, O.INSTALL	
FPT_RCV.3/Installer	O.OPERATE, O.RESOURCES, O.DELETION, O.INSTALL	
FDP_ACC.2/ADEL	O.DELETION	
FDP_ACF.1/ADEL	O.DELETION	
FDP_RIP.1/ADEL	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.DELETION	
FMT_MSA.1/ADEL	O.SID, O.FIREWALL, O.DELETION	
FMT_MSA.3/ADEL	O.SID, O.FIREWALL, O.DELETION	
FMT_SMF.1/ADEL	O.SID, O.FIREWALL, O.RESOURCES	
FMT_SMR.1/ADEL	O.FIREWALL, O.RESOURCES, O.DELETION	
FPT_FLS.1/ADEL	O.OPERATE, O.RESOURCES, O.ALARM, O.DELETION	

Security Functional Requirements	Security Objectives	Rationale
FDP_RIP.1/ODEL	O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT, O.TRANSACTION, O.OBJ-DELETION	
FPT_FLS.1/ODEL	O.OPERATE, O.RESOURCES, O.ALARM, O.OBJ-DELETION	
FCO_NRO.2/CM	O.LOAD	
FDP_IFC.2/CM	O.LOAD	
FDP_IFF.1/CM	O.LOAD	
FDP_UIT.1/CM	O.LOAD	
FIA_UID.1/CM	O.LOAD	
FMT_MSA.1/CM	O.SID, O.FIREWALL	
FMT_MSA.3/CM	O.SID, O.FIREWALL	
FMT_SMF.1/CM	O.SID, O.FIREWALL, O.RESOURCES	
FMT_SMR.1/CM	O.FIREWALL, O.RESOURCES	
FTP_ITC.1/CM	O.LOAD	
FPT_RCV.3/OS	O.SCP.RECOVERY, O.SCP.SUPPORT	
FPT_RCV.4/OS	O.SCP.RECOVERY, O.SCP.SUPPORT	
FPT_FLS.1/OS	O.SCP.IC, O.SCP.SUPPORT	
FPT_PHP.3/OS	O.SCP.IC, O.SCP.SUPPORT	
FDP_ACC.1/CardLifeCycleManagement	O.CARD-MANAGEMENT	
FDP_ACF.1/CardLifeCycleManagement	O.CARD-MANAGEMENT	
FMT_MSA.1/CardLifeCycleManagement	O.CARD-MANAGEMENT	
FMT_MSA.3/CardLifeCycleManagement	O.CARD-MANAGEMENT	
FTP_ITC.1/CardLifeCycleManagement	O.CARD-MANAGEMENT	
FCS_CKM.2/PACE	O.CIPHER, O.KEY-MNGT	
FCS_CKM.3/PACE	O.CIPHER, O.KEY-MNGT	
FCS_COP.1/PACE	O.CIPHER, O.KEY-MNGT	

Table 8 SFRs and Security Objectives

8.3.3 Dependencies

8.3.3.1 SFRs Dependencies

Requirements	CC Dependencies	Satisfied Dependencies
FDP_ITC.2/Installer	(FDP_ACC.1 or FDP_IFC.1) and (FPT_TDC.1) and (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM, FPT_TDC.1
FMT_SMR.1/Installer	(FIA_UID.1)	
FPT_FLS.1/Installer	No Dependencies	
FPT_RCV.3/Installer	(AGD_OPE.1)	AGD_OPE.1
FDP_ACC.2/ADEL	(FDP_ACF.1)	FDP_ACF.1/ADEL
FDP_ACF.1/ADEL	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.2/ADEL, FMT_MSA.3/ADEL
FDP_RIP.1/ADEL	No Dependencies	
FMT_MSA.1/ADEL	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/ADEL, FMT_SMF.1/ADEL, FMT_SMR.1/ADEL
FMT_MSA.3/ADEL	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/ADEL, FMT_SMR.1/ADEL
FMT_SMF.1/ADEL	No Dependencies	
FMT_SMR.1/ADEL	(FIA_UID.1)	
FPT_FLS.1/ADEL	No Dependencies	
FDP_RIP.1/ODEL	No Dependencies	
FPT_FLS.1/ODEL	No Dependencies	
FCO_NRO.2/CM	(FIA_UID.1)	FIA_UID.1/CM

Requirements	CC Dependencies	Satisfied Dependencies
FDP_IFC.2/CM	(FDP_IFF.1)	FDP_IFF.1/CM
FDP_IFF.1/CM	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.2/CM, FMT_MSA.3/CM
FDP_UIT.1/CM	(FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM
FIA_UID.1/CM	No Dependencies	
FMT_MSA.1/CM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_IFC.2/CM, FMT_SMF.1/CM, FMT_SMR.1/CM
FMT_MSA.3/CM	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/CM, FMT_SMR.1/CM
FMT_SMF.1/CM	No Dependencies	
FMT_SMR.1/CM	(FIA_UID.1)	FIA_UID.1/CM
FTP_ITC.1/CM	No Dependencies	
FPT_RCV.3/OS	(AGD_OPE.1)	AGD_OPE.1
FPT_RCV.4/OS	No Dependencies	
FPT_FLS.1/OS	No Dependencies	
FPT_PHP.3/OS	No Dependencies	
FDP_ACC.2/FIREWALL	(FDP_ACF.1)	FDP_ACF.1/FIREWALL
FDP_ACF.1/FIREWALL	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.2/FIREWALL, FMT_MSA.3/FIREWALL
FDP_IFC.1/JCVM	(FDP_IFF.1)	FDP_IFF.1/JCVM

Requirements	CC Dependencies	Satisfied Dependencies
FDP_IFF.1/JCVM	(FDP_IFC.1) and (FMT_MSA.3)	FDP_IFC.1/JCVM, FMT_MSA.3/JCVM
FDP_RIP.1/OBJECTS	No Dependencies	
FMT_MSA.1/JCRE	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FMT_SMR.1
FMT_MSA.1/JCVM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_SMF.1, FMT_SMR.1
FMT_MSA.2/FIREWALL_JCVM	(FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1
FMT_MSA.3/FIREWALL	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1
FMT_MSA.3/JCVM	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/JCVM, FMT_SMR.1
FMT_SMF.1	No Dependencies	
FMT_SMR.1	(FIA_UID.1)	FIA_UID.2/AID
FCS_CKM.1	(FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4)	FCS_CKM.2, FCS_CKM.4

Requirements	CC Dependencies	Satisfied Dependencies
FCS_CKM.2	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1, FCS_CKM.4
FCS_CKM.3	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1, FCS_CKM.4
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2)	FCS_CKM.1
FCS_COP.1	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	FCS_CKM.1, FCS_CKM.4
FDP_RIP.1/ABORT	No Dependencies	
FDP_RIP.1/APDU	No Dependencies	
FDP_RIP.1/bArray	No Dependencies	
FDP_RIP.1/KEYS	No Dependencies	
FDP_RIP.1/TRANSIENT	No Dependencies	
FDP_ROL.1/FIREWALL	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM
FAU_ARP.1	(FAU_SAA.1)	
FDP_SDI.2	No Dependencies	
FPR_UNO.1	No Dependencies	

Requirements	CC Dependencies	Satisfied Dependencies
FPT_FLS.1	No Dependencies	
FPT_TDC.1	No Dependencies	
FIA_ATD.1/AID	No Dependencies	
FIA_UID.2/AID	No Dependencies	
FIA_USB.1/AID	(FIA_ATD.1)	FIA_ATD.1/AID
FMT_MTD.1/JCRE	(FMT_SMF.1) and (FMT_SMR.1)	FMT_SMF.1, FMT_SMR.1
FMT_MTD.3/JCRE	(FMT_MTD.1)	FMT_MTD.1/JCRE
FDP_ACC.1/CardLifeCycleManagement	(FDP_ACF.1)	FDP_ACF.1/CardLifeCycleManagement
FDP_ACF.1/CardLifeCycleManagement	(FDP_ACC.1) and (FMT_MSA.3)	FDP_ACC.1/CardLifeCycleManagement, FMT_MSA.3/CardLifeCycleManagement
FMT_MSA.1/CardLifeCycleManagement	(FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1)	FDP_ACC.1/CardLifeCycleManagement, FMT_SMF.1/CM, FMT_SMR.1/CM
FMT_MSA.3/CardLifeCycleManagement	(FMT_MSA.1) and (FMT_SMR.1)	FMT_MSA.1/CardLifeCycleManagement, FMT_SMR.1/CM
FTP_ITC.1/CardLifeCycleManagement	No Dependencies	
FCS_CKM.2/PACE	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	

Requirements	CC Dependencies	Satisfied Dependencies
FCS_CKM.3/PACE	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	
FCS_COP.1/PACE	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4)	

Table 9 SFRs Dependencies

Rationale for the exclusion of Dependencies

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

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

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

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

The dependency FCS_CKM.1 of FCS_CKM.2/PACE, FCS_CKM.3/PACE and FCS_COP.1/PACE is discarded. The FCS_CKM.1 is implicit. Keys are generated with key distribution

The dependency FCS_CKM.4 of FCS_CKM.2/PACE, FCS_CKM.3/PACE and FCS_COP.1/PACE is discarded. The FCS_CKM.4 key destruction is operated by the calling application.

8.3.3.2 SARs Dependencies

Requirements	CC Dependencies	Satisfied Dependencies
ADV_ARC.1	(ADV_FSP.1) and (ADV_TDS.1)	ADV_FSP.5, ADV_TDS.4

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

Table 10 SARs Dependencies

8.3.4 Rationale for the Security Assurance Requirements

The chosen assurance level EAL5 and the augmentation with the requirements ALC_DVS.2 and AVA_VAN.5 were chosen in order to meet the assurance expectations for this type of TOE since it is intended to defend against highly sophisticated attacks without protective environment. This evaluation assurance package was selected to permit a developer to gain maximum assurance from positive security engineering based on good commercial practices. The augmentations are in compliance with the Protection Profile.

8.3.5 AVA_VAN.5 Advanced methodical vulnerability analysis

The selection of the component AVA_VAN.5 provides a higher assurance of the security by vulnerability analysis to assess the resistance to penetration attacks performed by an attacker possessing a high attack potential.

8.3.6 ALC_DVS.2 Sufficiency of security measures

Those requirements is the most adequate for a manufacturing process in which several actors (Platform Developer, Operator, Application Developers, IC Manufacturer, etc) exchange and store highly sensitive information (confidential code, cryptographic keys, personalisation data, etc).

9 TOE Summary Specification

9.1 Functionalities

F.OPEN

The **F.OPEN** subsystem provides the following functionalities:

- APDU dispatcher
- Application invocation

F.CARD_MANAGER

The **F.CARD_MANAGER** subsystem provides the following functionalities:

- Verification management
- Load management
- Install management
- Issuer Security Domain
- Supplementary Security Domain

F.JAVA_CARD_SYSTEM

The **F.JAVA_CARD_SYSTEM** subsystem provides the following functionalities:

- Virtual machine
- Runtime environment
- Loader Linker
- Garbage collector

F.JAVA_API

The **F.JAVA_API** subsystem provides the following functionalities:

- GlobalPlatform API
- Security and cryptography framework (javacard.security API)
- Java Card Framework (javacard.framework API)
- Biometric framework (javacardx.biometry API)
- Security and cryptography extension (javacardx.security API)
- External memory access (javacardx.external API)
- Extension framework (javacardx.framework API)
- SAC protocol (com.morpho.sac API)
- SM accelerator (com.morpho.sm API)

F.AUTHENTICATION

The **F.AUTHENTICATION** subsystem provides the following functionalities:

- Supplementary access control establishment
- Secure communication management

This sub-system comes in support to **F.CARD_MANAGER**

F.CRYPTOGRAPHY_SERVICES

The **F.CRYPTOGRAPHY_SERVICES** subsystem provides the following functionalities:

- Key protection (in non-volatile memory)

- Random number generation
- Hash computation
- Data ciphering and deciphering
- CRC computation
- Symmetric and asymmetric signature
- Elliptic curves diffe-hellman key agreement
- Key derivation and generation
- Secure channel protocol
- Crypto self-tests

This sub-system comes in support to **F.JAVA_API**

F.SECRET_DATA_MANAGER

The **F.SECRET_DATA_MANAGER** subsystem provides the following functionalities:

- Biometric authentication algorithms

F.SECURE_DATA_MANAGER

The **F.SECURE_DATA_MANAGER** subsystem provides the following functionalities:

- PIN management
- Key management

This sub-system comes in support to **F.JAVA_API**

F.SYSTEM_MANAGER

The **F.SYSTEM_MANAGER** subsystem provides the following functionalities:

- System initialization
- Configuration dispatcher
- Configuration parameters
- Non volatile memory protection management
- Application registry

F.MEMORY_ACCESS

The **F.MEMORY_ACCESS** subsystem provides the following functionalities:

- Non volatile memory allocation
- Java objects access

F.MEMORY_PROGRAMMING

The **MEMORY_PROGRAMMING** subsystem provides the following functionalities:

- Transaction and atomicity management
- Dynamic RAM allocation

F.INPUT/OUTPUT_LAYER

The **F.INPUT/OUTPUT_LAYER** subsystem provides the following functionalities:

- Communication initialization
- Communication exchange
- I/O state management
- ATR management

- APDU buffer management

F.CRYPTOGRAPHIC_OPERATIONS

The **F.CRYPTOGRAPHIC_OPERATIONS** subsystem provides the following functionalities:

- DES algorithm operations
- AES algorithm operations
- SHA algorithm operations
- Modular exponentiation operations
- Elliptic curves operations
- Key protection (in volatile memory)
- Utilities
- Random number generation
- XRC operations

F.MEMORY_CONTROLLER

The **F.MEMORY_CONTROLLER** subsystem provides the following functionalities:

- NVM writing
- Memory copy and compare operations

F.TRANSPORT_LAYER

The **F.TRANSPORT_LAYER** subsystem provides the following functionalities:

- Interface availability control
- Time extension management
- Contact interface management
- Contactless interface management

F.CPU_MANAGER

The **F.CPU_MANAGER** subsystem provides the following functionalities:

- CPU serial number access
- CPU configuration access

F.SECURITY_CONFIGURATION

The **F.SECURITY_CONFIGURATION** subsystem provides the following functionalities:

- Card security configuration
- Card security operations

F.SECURITY_AUDIT

The **F.SECURITY_AUDIT** subsystem provides the following functionalities:

- Security audit and reaction operation

F.CRYPTOGRAPHIC_LIBRARY

The **F.CRYPTOGRAPHIC_LIBRARY** subsystem provides the following functionalities:

- Cryptographic routines

F.INTEGRATED_CIRCUIT

The **F.INTEGRATED_CIRCUIT** subsystem provides the following functionalities:

- Physical protection mechanisms

10 Appendix

10.1 Definitions

This section provides definitions about terms frequently used in this document. The definition of the Common Criteria related terms is specified in **[CC_Part1]**.

10.2 Abbreviations

Acronym	Definition
AES	Advanced Encryption Standard
AID	Application IDentifier
APDU	Application Protocol Data Unit
ATR	Answer To Reset
CAD	Card Acceptance Device
CBC	Cipher Block Chaining
CC	Common Criteria
CRT	Chinese Remainder Theorem
DES	Data Encryption Standard
EAL	Evaluation Assurance Level
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptosystem
ECDH	Elliptic Curve Diffie Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
GP	GlobalPlatform
IC	Integrated Circuit(s) Card
JCAPI	JavaCard Application Programming Interface
JCRE	JavaCard Runtime Envrironment
JCVM	JavaCard Virtual Machine
OSP	Organizational Security Policy
PM	Project Manager
PP	Protection Profile
RNG	Random Number Generator
RSA	Rivest Shamir Adleman
SAR	Security Assurance Requirements
SCP	Smart Card Platform
SCP02	Secure Channel Protocol 02
SCP03	Secure Channel Protocol 03
SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SPD	Security Problem Definition

Acronym	Definition
ST	Security Target
TOE	Target Of Evaluation
TSF	TOE Security Functions
VA	Verification Authority

10.3 References

Ref.	Document title
[CC_Part1]	"Common Criteria for information Technology Security Evaluation, Part 1: Introduction and general model", Version 3.1 – Revision 4, September 2012
[CC_Part2]	"Common Criteria for information Technology Security Evaluation, Part 2: Security Functional Requirements", Version 3.1 – Revision 4, September 2012
[CC_Part3]	"Common Criteria for information Technology Security Evaluation, Part 2: Security Assurance Requirements", Version 3.1 – Revision 4, September 2012
[CCDB_COMP]	CCDB, Composite product evaluation for Smart Cards and similar devices, CCDB-2012-04-001, Version 1.2 - Revision 1, April 2012,
[GP_AmdtD]	GlobalPlatform Card Technology – Secure Channel Protocol 03 – Card Specification v 2.2 – Amendment D Version 1.1, September 2009
[GP_CS]	GlobalPlatform, Card Specification, Version 2.1.1, March 2003
[IEEE1363]	IEEE Standard Specifications for Public-Key Cryptography, 30 January 2000
[JCAPI]	JavaCard Platform, versions 3.0 (March 2008) and 3.0.1, Classic Edition, including Specification Errata, October 2010, Updated February 2011, Application Programming Interface, March 2008. Published by Sun Microsystems, Inc.
[JCVM]	JavaCard Platform, version 3.0.1, Classic Edition, including Specification Errata, October 2010, Updated February 2011 Virtual Machine (JavaCard VM) Specification. Published by Sun Microsystems, Inc.
[JCRE]	JavaCard Platform, version 3.0.1 (, Classic Edition, including Specification Errata, October 2010, Updated February 2011. Runtime Environment (JavaCard RE) Specification. March 2008. Published by Sun Microsystems, Inc.
[PP_IC]	Security IC Platform Protection Profile, Version 1.0, 15.06.2007 as certified under BSI-PP-0035
[PP_JC]	JavaCard Protection Profile – Open Configuration, Version 3.0, May, 2012. Certified by ANSSI under the reference ANSSI-CC-PP-2010/03-M01
[PP_USIM]	"(U)SIM Java Card Platform Protection Profile – Basic and SCWS Configurations", PU-2009-RT-79, version 2.0.2

Ref.	Document title
[RGS_B1]	"Mécanismes Cryptographiques – Règles et Recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques", Référentiel General de Security, Annexe B1, Version 1.20, 26 Janvier 2010
[ST_IC]	Security Target Lite M7892 B11 including optional software libraries RSA – EC – SHA-2 - Toolbox, Version 1.1, 2012-08-28
[NOTE_ANSSI]	Note d'application ANSSI-CC-NOTE-10-0, Certification d'application sur « Plateformes ouvertes cloisonnantes »
[ICAO-SAC]	International Civil Aviation Organization, ICAO MACHINE READABLE TRAVEL DOCUMENTS, TECHNICAL REPORT, Supplemental Access Control for Machine Readable Travel Documents, Version 1.00, November 2010

Index