# Public Security Target
# ID-ONE COSMO X
# EAL6+

**Edition: 1**

## DOCUMENT EVOLUTION

| Issue | Date | Author | Purpose |
|---|---|---|---|
| 1 | 02/06/2023 | IDEMIA | Public ST created from Full ST FQR 110 9792 Ed 5 |

# Table of contents

# Table of figures

# Table of tables

# 1 SECURITY TARGET INTRODUCTION

This Public Security Target aims to satisfy the requirements of Common Criteria level EAL6+, augmented with ALC_FLR.1 in defining the security enforcing functions of the Target Of Evaluation and describing the environment in which it operates.
The basis for this composite evaluation is the composite evaluation of Platform and the hardware plus the cryptographic library.

**Application note**
AVA_VAN.5 implies that the TOE is resistant to attacks performed by an attacker possessing "High Attack Potential".
Not all key sizes specified in this security target have sufficient cryptographic strength for satisfying the AVA_VAN.5 "High Attack Potential". In order to be protected against attackers with a "High Attack Potential", sufficiently large cryptographic key sizes SHALL be configured for this TOE. Please refer to national and international document standards for more and up-to-date details.

## 1.1 Public Security Target Reference

| | |
|---|---|
| **Title** | Public Security Target ID-One Cosmo X –EAL6+ |
| **Editor** | IDEMIA |
| **IDEMIA registration** | FQR 110 A23D |
| **Revision** | Ed 1 |
| **ITSEF** | SGS-BS |

## 1.2 TOE Reference

| | |
|---|---|
| **Product Commercial Names** | ID-One Cosmo X Platform |
| **IC** | Certificate number: certified by the German BSI certification body under code: BSI_DSZ-CC-1107-V3-2022 |
| **TOE Name** | ID-One Cosmo X |
| **version (R3 + patch)** | SAAAAR Code: 093363<br>SAAAAR Code: 099E71 |
| **version (R4 + 2 patchs)** | SAAAAR Code: 093364<br>SAAAAR Code: 099441<br>SAAAAR Code: 099E21 |
| **Version (R6)** | SAAAAR Code: 093366 |
| **TOE Documentation** | Refer to Section 1.7.3 TOE Guidance |

The TOE and the product differ, as further explained in Architecture of the product:
The TOE is the Java Card System (JCS) Open Platform of the ID-One Cosmo product and the product may also include applets.

The TOE is identified by the tag identity which provides information on the product and allows identifying each product configuration in term of features included or not. For instance, the configuration for the embedded modules with FLEXICODE can be retrieved. Information and values to identified TOE are described in **[AGD_OPE]**.

## 1.3 Security Target Overview

The main objectives of this Security Target are to:

- Introduce the JCS Platform,
- Define the scope of the TOE and its security features
- Describe the security environment of the TOE, including the assets to be protected and the threats to be countered by the TOE and its environment during the product development, production and usage.
- Describe the security objectives of the TOE and its environment supporting in terms of integrity and confidentiality of application data and programs and of protection of the TOE.
- Specify the security requirements, which include the TOE security functional requirements, the TOE assurance requirements and TOE security functions.

## 1.4 REFERENCES

| Common Criteria Ref. | Document details |
|---|---|
| **[CC1]** | "Common Criteria for information Technology Security Evaluation, Part 1: Introduction and general model", April 2017, Version PP3.1 revision 5. |
| **[CC2]** | "Common Criteria for information Technology Security Evaluation, Part 2: Security Functional component", April 2017, Version 3.1 revision 5. |
| **[CC3]** | "Common Criteria for information Technology Security Evaluation, Part 3: Security Assurance components", April 2017, Version 3.1 revision 5. |
| **[CEM]** | «  Common Methodology for Information Technology Security Evaluation »  Evaluation methodology April 2017 Version 3.1 Revision 5 |
| **[NOTE6]** | Note d'application exigences de sécurité pour un chargement de code en phase d'utilisation-  ANSSI-CC-NOTE-06/2.0, or<br>JIL Security requirements Library for post-delivery code loading, Version 1.0 February 2016. |
| **[NOTE10]** | Certification of « open » smart card products, Version 0.1 (for trial use), 27 July 2012. |
| **[COMPO]** | "Composite product evaluation for Smart Cards and similar devices"<br>May 2018, Version 1.5.1 |
| **[JIL1]** | JIL-Guidance-for-smartcard-evaluation-v2-0 |
| **[PP0099]** | Java Card System – Open Configuration Protection Profile, Version 3.0.5 December 2017, BSI-CC-PP-0099-2017 |
| **[PP0068]** | Protection Profile Machine Readable Travel Document using Standard Inspection Procedure with PACE, BSI-CC-PP-0068-V2-MA-01, Version 1.0.1, 22 July 2014, BSI. |
| **[PP0084]** | Security IC Platform Protection Profile with Augmentation Packages Version 1.0, 13 January 2014, BSI-CC-PP-0084-2014 |
| **[PP0035]** | Java Card System Standard 2.2 Configuration Protection Profile – PP/0305 Version 1.0b – August 2003 |
| **[IFX_ST]** | IFX_CCI_00002Dh, IFX_CCI_000039h, IFX_CCI_00003Ah, IFX_CCI_000044h, IFX_CCI_000045h, IFX_CCI_000046h, IFX_CCI_000047h, IFX_CCI_000048h, IFX_CCI_000049h, IFX_CCI_00004Ah, IFX_CCI_00004Bh, IFX_CCI_00004Ch, IFX_CCI_00004Dh, IFX_CCI_00004Eh T11 Security Target Lite, Infineon Technologies AG, Version 4.3.1, 2022-05-10 |

| IDEMIA Ref | Document details |
|---|---|
| **[AGD_SR]** | ID-One Cosmo X Applet Security Recommendations, FQR 110 9572 ED7 |
| **[AGD_OPE]** | ID-One Cosmo X Reference Guide, FQR 110 9563 ED13 |
| **[AGD_PRE]** | ID-One Cosmo X Pre-Perso Guide, FQR 110 9562 ED13 or ED14<br>The difference between Ed13 and ED 14 is editoriale |
| **[AGD_ALP]** | ID-One Cosmo X Application Loading Protection Guidance, FQR 110 9603 ED3 |
| **[AGD_PAPI]** | ID-One Cosmo X Javadoc, FQR 110 9616 ED4 |
| **[AGD_BIO]** | Biometry on ID-One Cosmo X (SLC37) , FQR 110 9598 ED2 |
| **[JPATCH]** | JCVM Patch, FQR 110 8805 ED4 |
| **[JBOX]** | JBox SW Configuration, FQR_110_9273 ED2 |
| **[GPP]** | Global Privacy Framework, FQR 110 9567 ED3 |
| **[GEN]** | IDEMIA Platform Flash Generation, FQR 110 9402 ED1 |
| **[DEL]** | Secure acceptance and delivery of sensitive elements, FQR 110 8921 ED1 |

| External Spec. | Document details |
|---|---|
| **[JCAPI]** | "Java Card 3.1 Classic - API"<br>Application Programming Interfaces, Version 3.1, 2019, Oracle Technology Network |
| **[JCRE]** | Java Card Platform Runtime Environment Specification, Classic Edition Version 3.1 November 2019, Oracle Technology Network |
| **[JCVM]** | Java Card Platform<br>Virtual Machine Specification, Classic Edition Version 3.1 November 2019, ,<br>Oracle Technology Network |
| **[JVM]** | The Java Virtual Machine Specification. Lindholm, Yellin<br>ISBN 0-201-43294-3 |
| **[JCOCV]** | Java Card 3 Platform Off-card Verification Tool Specification, Classic Edition, Version 1.0. Published by Oracle |
| **[GP1]** | "GlobalPlatform Card Specification"<br>Version 2.3.1 Public Release - March 2018<br>Document Reference: GPC_SPE_034 |
| **[GP2]** | GlobalPlatform Card Technology - Secure Channel Protocol '03', Card Specification v2.3 – Amendment D"<br>Version 1.1.2 - Public Release March 2019<br>Document Reference: GPC_SPE_014 |
| **[GP3]** | GlobalPlatform Government Task Force -Privacy Framework Requirements<br>Version 1.0 January 2013<br>Document Reference: GP_REQ_010 |
| **[GP4]** | GlobalPlatform Card Common Implementation Configuration Version 2.1<br>July 2018 Document Reference: GPC_GUI_080 |
| **[ISO1]** | "Identification cards - Integrated Circuit(s) Cards with contacts, Part 6: Interindustry data elements for interchange"<br>ISO/IEC 7816-6 (2004) |
| **[DSA]** | "Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)"<br>ANSI X9.31-1998, American Bankers Association |
| **[FIPS1]** | "FIPS PUB 46-3, Data Encryption Standard"<br>October 25, 1999 (ANSI X3.92), National Institute of Standards and Technology |
| **[FIPS_DES]** | "FIPS PUB 81, DES Modes of Operation"<br>April 17, 1995, National Institute of Standards and Technology |

| External Spec. | Document details |
|---|---|
| **[FIPS_HASH]** | "FIPS PUB 180-3, Secure Hash Standard"<br>October 2008 , National Institute of Standards and Technology |
| **[FIPS_DSS]** | "FIPS PUB 186-3"<br>June 2009, Digital Signature Standard (DSS) |
| **[FIPS_AES]** | FIPS PUB 197, The Advanced Encryption Standard (AES)<br>U.S. DoC/NIST, November 26, 2001. |
| **[FIPS 140-3]** | "FIPS PUB 140-3, Security requirements for cryptographic modules"<br>March 22, 2019, National Institute of Standards and Technology |
| **[ECDSA]** | ANSI x9.62-2005 Public Key Cryptography for the Financial Services Industry – The Elliptic Curve Digital Signature Algorithm (ECDSA) |
| **[RSA]** | PKCS#1 RSA Cryptography standards v2.2 27th october 2012 |
| **[ISO2]** | "Public Key Cryptography using RSA for the financial services industry"<br>ISO/IEC 9796-1, annex A, section A.4 and A.5, and annex C (1995) |
| **[ISO3]** | "Information technology – Security techniques: Data integrity mechanism using a cryptographic check function employing a block cipher algorithm"<br>ISO/IEC 9797-1 (1999) , International Organization for Standardization |
| **[IEEE]** | IEEE Std 1363a-2004 Standard Specification of Public-Key Cryptography |
| **[NIST_RNG]** | The NIST SP 800-90 Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revise)<br>March 2007 |
| **[ICAO-9303]** | 'ICAO Doc 9303', Machine Readable Travel Documents, Seventh Edition, 2015 – Security Mechanisms for MRTDs |
| **[PKI]** | Machine Readable Travel Documents Technical Report, PKI for Machine Readable Travel Documents Offering ICC Read-Only Access, Version - 1.1, Date - October 01, 2004, published by authority of the secretary general, International Civil Aviation Organization |
| **[ICAO-TR]** | International Civil Aviation Organization, ICAO MACHINE READABLE TRAVEL DOCUMENTS, TECHNICAL REPORT, Supplemental Access Control for Machine Readable Travel Documents, Version 1.1, April 2014 |
| **[BRAINPL]** | Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation (RFC 5639), Manfred Lochter and Johannes Merkle, IETF Trust, March 2010 |
| **[TR_03110]** | Technical Guideline TR-03110-1, Advanced Security Mechanisms for Machine Readable Travel Documents –Part 1 – eMRTDs with BAC/PACEv2 and EACv1, Version 2.2, Bundesamt für Sicherheit in der Informationstechnik (BSI), 26.02.2015 |

# 1.5 ACRONYMS and NOTATIONS

## *1.5.1 Abbreviations*

| | |
|---|---|
| AES | Advanced Encryption Standard |
| AID | Applet Identifier |
| APDU | Application Protocol Data Unit |
| API | Application Programmer Interface |
| BIOS | Basic Input/Output System |
| CC | Common Criteria |
| CM | Card Manager |
| CLFDB | Ciphered Load File Data Block |
| CPLC | Card Production Life Cycle |
| DAP | Data Authentication Pattern |
| DBI | Digitally Blurred Image |
| DES | Cryptographic module "Data Encryption Standard" |
| DSK | Dump Secret Key |
| EAL | Evaluation Assurance Level |
| EC | Elliptic Curves |
| ECC | Elliptic Curve Cryptography |
| GP | Global Platform |
| IC | Integrated Circuit |
| ISD | Issuer Security Domain |
| IT | Information Technology |
| JCP | Java Card Platform |
| JCRE | Java Card Runtime Environment |
| ISK | Initialization Secret Key |
| JSK | JPatch Secret Key |
| LSK | Load Secret Key |
| MOC | Match-On-Card |
| MRTD | Machine readable travel document |
| MSK | Master Secret Key |
| NA | Not Applicable |
| OSP | Organizational Security Policy |
| PACE | Password Authenticated Connection Establishment |
| PP | Protection Profile |
| PUK | Personal Unlocking Key |
| RNG | Random Number Generation |
| RSA | Cryptographic module "Rivest, Shamir, Adleman" |
| SF | Security Function |
| SFP | Security Function Policy |
| SHA-1 | Cryptographic module "Secure hash standard" |
| ST | Security Target |
| TOE | Target of Evaluation |
| TSF | TOE Security Functions |
| VM | Virtual Machine |

### 1.5.2 Definitions

| | |
|---|---|
| Applet | Application which can be loaded and executed with the environment of the Java Card platform. |
| Applet developer | The one who is in charge of the Applet development intended to be loaded on the platform. He/she is the recipient of the in order to respect recommendations, if any, identified by the platform evaluation. These recommendations shall be followed by the applet developer and shall be checked before loading the application on the platform. |
| Card Issuer | Entity that owns the card and is ultimately responsible for the behaviour of the card. |
| Card Manager | Main entity which represents the issuer and supervises the whole services available on the card. The Card Manager entity encompasses the Open and the Issuer Security domain. |
| Cipher Load File Data Block | The 'CLFDB' is to be considered when the encryption of Load File Data Block is required. This privilege allows an SD Provider to require ciphering the Load File Data Block. The SD who has this privilege will be requested by the TOE to decrypt the Load File Data Blocks and their associated Executable Load Files. |
| DAP | Part of the Load File used for ensuring authenticity of the Load File. The DAP is the signature of the Load File Data Block Hash and is provided during the loading. |
| DSK | Secret key used to read/write the personalization configuration dumped. |
| Global Privacy Protocol | Privacy requirements that apply to an entire platform (and which may be the only privacy requirements defined for certain applications). |
| Issuer Security Domain | The primary on-card entity providing support for the control, security, and communication requirements of the Card Issuer. |
| JSK | Secret key used for patch loading. |
| Local or Specific Privacy Protocol | Privacy requirements that apply to a particular application; may be further limited as applying only to a particular physical interface and/or life cycle state. |
| Load File Data Block Hash or LoadFile | The Load File Data Block Hash provides integrity of the Load File Data Block following receipt of the complete Load File Data Block. |
| LSK | Secret Key used by the OS developer (the TOE developer) to encrypt locks and patches. |
| Machine readable travel document (MRTD) | Official document issued by a State or Organization which is used by the holder for international travel (e.g. passport, visa, official document of identity) and which contains mandatory visual (eye readable) data and a separate mandatory data summary, intended for global use, reflecting essential data elements capable of being machine read. **[ICAO-9303]** |
| Match On Card | The match on card (MOC) technology consists in registering the face, iris and fingerprint biometrics (or their template) on a smart card or a USB key, the card being unlocked using the finger, face or Iris that functions as code. |
| MSK | Master Secret Key for authentication used to authenticate the card Manufacturer. This key has a given value (i.e. MSK value) and its try counter, i.e. MSK Key counter (as for a PIN). Once the try limit counter is reached, the authentication is no more possible with this key. |
| OPEN | Part of the Card Manager entity which has the responsibilities to provide an API to applications, command dispatch, Application selection, logical channel management, and Card Content management. The OPEN also manages the |

| | |
|---|---|
| | installation of applications loaded to the card. The OPEN is responsible for enforcing the security policy defined for Card Content management. |
| PACE | A communication establishment protocol defined in **[ICAO-9303]** part 11. The PACE Protocol is a password authenticated Diffie-Hellman key agreement protocol providing implicit password-based authentication of the communication partners (e.g. smart card and the terminal connected): i.e. PACE provides a verification, whether the communication partners share the same value of a password π). Based on this authentication, PACE also provides a secure communication, whereby confidentiality and authenticity of data transferred within this communication channel are maintained. |
| PACE passwords | Passwords used as input for PACE. This may either be the CAN or the SHA-1-value of the concatenation of Serial Number, Date of Birth and Date of Expiry as read from the MRZ, see **[ICAO-9303]** part 11.<br>In addition, the passwords used to establish the PACE can be the PIN from the javacard standard or PUK from mobile phone standard, PIN and PUK come from applets. |
| PACE Terminal | A technical system verifying correspondence between the password stored in the electronic document and the related value presented to the terminal by the electronic document presenter. A PACE terminal implements the terminal part of the PACE protocol and authenticates itself to the electronic document using a shared password (CAN, eID-PIN, eID-PUK or MRZ). A PACE terminal is not allowed reading sensitive user data. |
| PUK | A long secret password being only known to the electronic document holder. A personal unlocking key, sometimes called personal unblocking code (PUC), is used to reset a personal identification number (PIN) that has been lost or forgotten. |
| Security Domain | On-card entity providing support for the control, security, and communication requirements of an off-card entity (e.g. the Card Issuer, an Application Provider or a Controlling Authority). |

# 1.6 TOE OVERVIEW

## *1.6.1 TOE Type*

The ID-One Cosmo X Platform on Infineon (IFX) chip is a dual Java Card platform based, compatible with multi-application ID-One Cosmo product family.

The functional level of the OS is based on a Java™ based multi-application platform, compliant with Java Card 3.1 Classic Edition and Global Platform 2.3 specifications.

This ID-One Cosmo X platform is able to receive and manage different types of applications, Basic and Sensitive ones.
All the platform code including GP Java application called card manager are loaded in the FLASH memory.

The TOE allows the suppression of Modules during pre-personalization or personalization with FLEXICODE mechanism.

The TOE allows the loading of optional code, Javacard applications and native code:

- Optional code can be loaded to upgrade the TOE (platform) at any time of product life cycle, this function is named JPatch.
- Native code, viewed as a library behind the JCS, can also be loaded at any time with the JBOX functionality.
- Applications can be loaded on the flash memory, at pre-personalisation, personalisation or use phase.
- Optional code (CodOp) can be loaded to upgrade an applet at any time of product life cycle.

However, the Card Issuer can forbid each of these operations before or after the issuance of the card.
The mechanism for the different loading are part of the present ST and part of the TOE evaluation.

## *1.6.2 TOE usage*

This Platform is an open and isolating platform that is compliant with the ANSSI Application Note 10 that deals with open and isolating platforms.
Smart cards are used as data carriers that are secure against forgery and tampering as well as personal, highly reliable, small size devices capable of replacing paper transactions by electronic data processing. Data processing is performed by a piece of software embedded in the smart card chip, called an application.
The Java Card System is intended to transform a smart card into a platform capable of executing applications written in a subset of the Java programming language. The intended use of a Java Card platform is to provide a framework for implementing IC independent applications conceived to safely coexist and interact with other applications into a single smart card.
The Platform is embedded with flexible code (FLEXICODE Mechanism): the platform can be configured by deleting modules when associated functionalities are not necessary for the intended usage.
Applications installed on a Java Card platform can be selected for execution when the card communicates with a card reader.
The communication can be established with the PACE protocol and associated secure messaging.

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

- Electronic passports and identity cards.
- Secure information storage, like health records, or health insurance cards.
- Financial applications, like Credit/Debit ones, stored value purse, or electronic commerce, among others.
- Transport and ticketing, granting pre-paid access to a transport system like the metro and bus lines of a city.
- Telephony, through the subscriber identification module (SIM) or the NFC chip for mobile phones.
- Personal identification, for granting access to secured sites or providing identification credentials to participants of an event.
- Loyalty programs, like the "Frequent Flyer" points awarded by airlines. Points are added and deleted from the card memory in accordance with program rules. The total value of these points may be quite high and they must be protected against improper alteration in the same way that currency value is protected.

Furthermore, this platform embeds the Biometric MOC algorithm, which is a highly secure technology for smart cards applications. The MOC technology is an entire part of the product and enables the authentication by way of digital prints.

# 1.7 PRODUCT ARCHITECTURE

## *1.7.1 Logical scope of the TOE*

The TOE is composed of hardware and software components, as listed below and described in Figure 1.

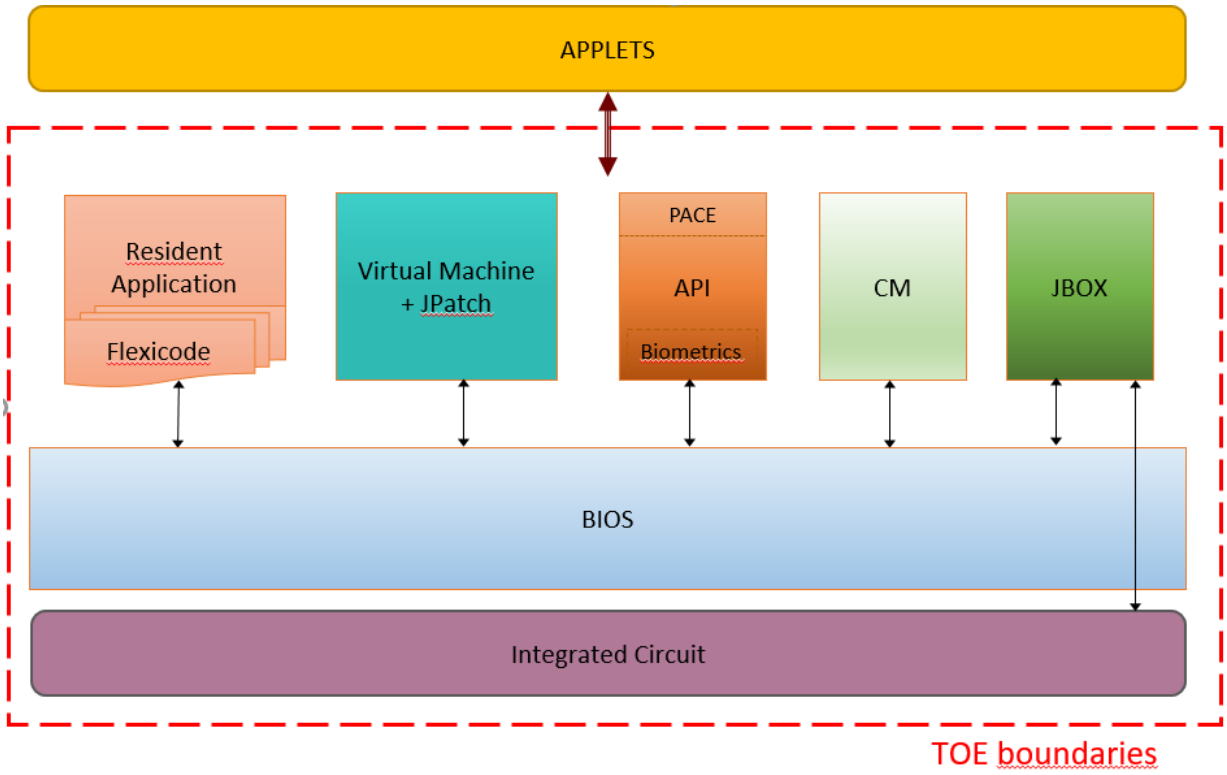


**Figure 1: Java Platform Architecture**

The TOE, boundaries defined by dotted red line, includes the BIOS, the JBox, the Virtual Machine, the APIs, the Global Platform application (with the CM), the Resident application, the PACE, the BIO and the IC component. The TOE integrates also a patch mechanism called Jpatch, implemented in the VM block. Details of components are presented in the TOE description.
This platform is a multi-modules architecture with flexible code (FLEXICODE), implemented in the RA block. Modules can be configured or deleted in order to free memory space in Flash.
The constitution of this modularity depends on customer need; during Cosmo personalization the FLEXICODE allows modules deletion. The platform construction ensures the isolation of native modules. The modules are all defined in the current security target that become also modular: When a module is still present, the associated SPD, the security objectives, the SFRs and the TSS define the TOE security. when a module is deleted from the TOE, the corresponding TSF is not supported.

### 1.7.2 Physical scope of the TOE

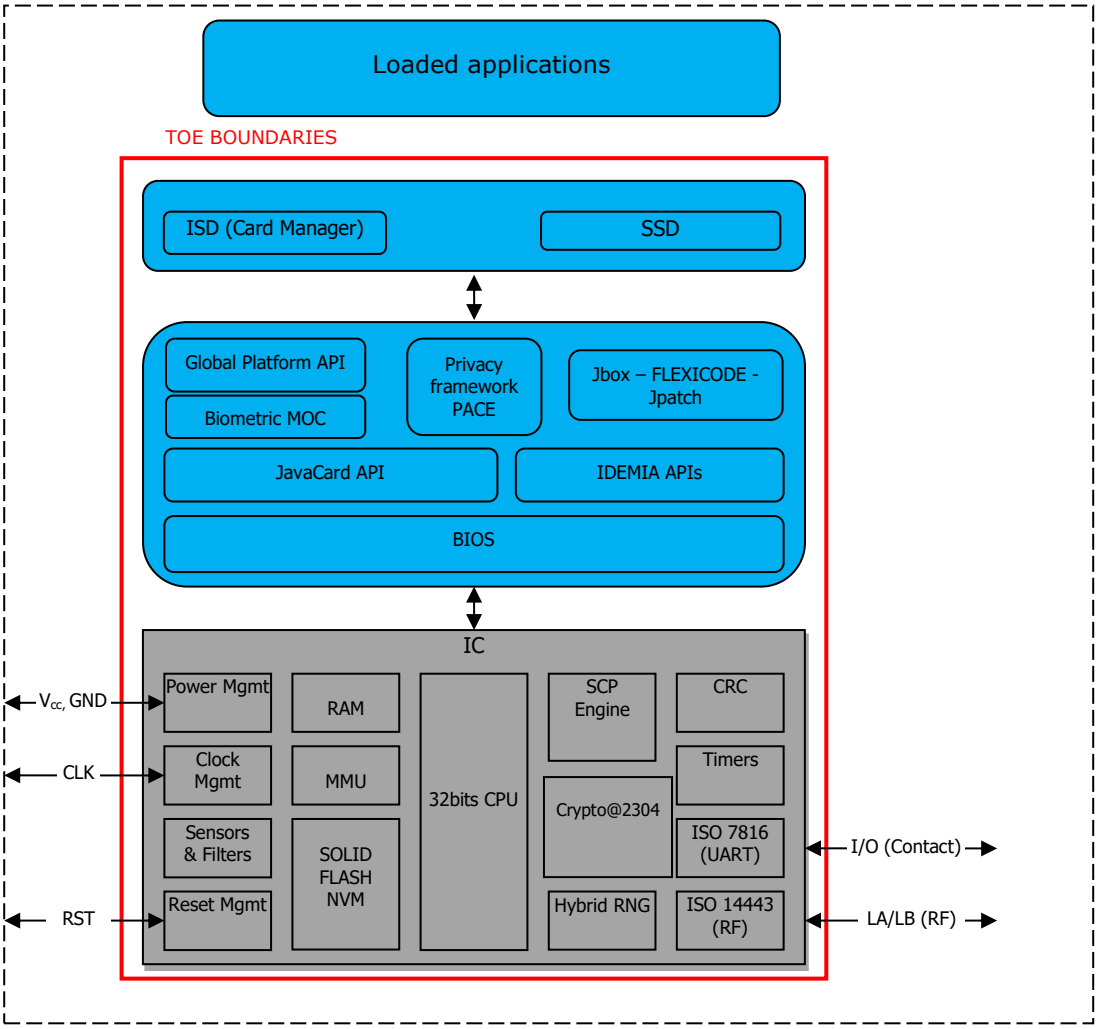


**Figure 2: View of the smart card and pins**

**The ID-One Cosmo X** on IFX is a dual Java Card platform based, compatible with multi-application ID-One Cosmo product family.
The TOE is defined by:

- The underlying IC with its dedicated software,
- the OS based on a Java™ based multi-application platform, compliant with **Java Card 3.1 Classic Edition** and **Global Platform 2.3** specifications.
- The ability to receive and manage different types of applications, Basic and Sensitive ones.
- All the platform code including GP Java application called card manager are loaded in the FLASH memory.
- The product is open at use phase.
- The basic or sensitive applications can be loaded on the flash memory, at prepersonalisation, personalisation or use phase. These applications are out or the scope of the present evaluation.

The Applets loaded pre issuance or post issuance are outside the TOE, Other smart card product elements, (such as holograms, inlays, security printing) are outside the scope of this Security Target.
Java Card RMI is not implemented in the TOE.

### 1.7.3 TOE Guidance

The ID-One Cosmo X is evaluated with its guidance. This Public ST derived from the restricted ST is also a guidance to all users of the TOE.
The guidance's of the Platform are listed hereafter:

| Audience | Ref | Form factor of delivery |
|---|---|---|
| Guidance of developer of sensitive applications | **[AGD_SR]** | Electronic version |
| Guidance for application developer | **[AGD_OPE]** **[AGD_PAPI]** | |
| Guidance for application developer with Biometry | **[AGD_BIO]** | |
| Guidance Platform Flash Generation | **[GEN]** | |
| Guidance for developer of patches using JPATCH and patch loading | **[JPATCH]** | |
| Guidance for developer of Native Code using JBOX and Code loading | **[JBOX]** | |
| Guidance for developer with GPP and LPP services | **[GPP]** | |
| Guidance for pre-personalisation | **[AGD_PRE]** | |
| Issuer of the platform that aims to load applications | **[AGD_ALP]** | |
| Secure acceptance | **[DEL]** | |

**Table 1: Guidance references**

**[AGD_SR]**
If the applet needs to have a security certification, the applet must follow recommendations listed in the document.
If the applet is a basic application, and does not need security certification with the platform, the certificate of the Platform is still valid if the applet go through the verifier before the loading of this applet (the security function of the platform are still available).
This guide is provided to the developer and evaluator of a sensitive application to be certified.

**[AGD_OPE]**

This document describes the ID-One Cosmo X smart card usage. It describes how to use the card from an APDU commands point of view and gets onto topics such as common platform APDU commands, secure channels and security domains.
This document also describes the available Java Card and proprietary APIs for applet developers.
This guide is provided to the users, personalizer and developer of Java Card applications to be certified or not. It does not mandate any requirement for the developer; it constitutes a help.

**[AGD_PRE]**
This document describes the pre-personalisation steps that should be followed to correctly initialize the ID-One Cosmo X platform. The TOE is finalized once it is pre-personalised.

**[AGD_ALP]**
This document describes the loading procedure, in compliance with **[NOTE10]** and the Java Card Open Platform protection profile.
The **[JPATCH]** is provided to the entity that has the card content management rights

**[AGD_PAPI]**
This document summarizes the ID proprietary API (packages, classes, methods and fields) available on the Java Card Identity Platforms.
This guide is provided to the Developer of Java Card applications to be certified or not. It does not mandate any requirement for the developer; it constitutes a help.

**[AGD_BIO]**
This document describes the software biometrics features for the platform, how to personalize and use it. This product embeds MOC biometric algorithms that have been develop by IDEMIA.

**[GEN]**
This document specifies the guidance to correctly load the platform code using the IC loader.

**[DEL]**
This document specifies the secure acceptance of the components that comprise the TOE.

**[JPATCH]**
Guidance for developer of patches using JPATCH and patch loading
The Guidance is aimed to be used by IDEMIA R&D. The patch has to be developed only by IDEMIA R&D. Any patch has to be evaluated:
- by maintenance process if the patch does not impact the security
- or by reassement if the intended patch impacts the security of any of evaluated security function of the present scope.

**[JBOX]**
Guidance for developer of Native Code using JBOX and Code loading
This guidance expresses the available interfaces for the developer of additional code required by the issuer. What ever is the code, the security and the security certificate of the platform are not reconsidered.

### *1.7.4 Platform isolation*

To ensure the platform isolation, the following verifications must be done:

1. For library packages intended to be loaded on the platform, the versioning rules must be applied: described in the Java Card Virtual Machine Specification at chapter "Binary Compatibility" and chapter "Package Version". In particular, this allows to determine the binary compatibility of this package with another previous version package. These rules are also summarized in "GlobalPlatform Card Composition Model Security Guidelines for Basic Applications" at chapter "Versioning".

2. The byte code verification is mandatory for non-evaluated applications. It must be done using export files provided by IDEMIA.
    a. For dependencies to standard packages, or those developed by IDEMIA verification must be done using the corresponding export files provided by IDEMIA.

Those verifications shall be done for Post-issuance personalisation of Security Domain or all application intended to be loaded on the platform.

### *1.7.5 Applications*

For sensitive application, the recommendations listed in **[AGD_SR]** are mandatory. The evaluator of the sensitive application, checks that the guidance is followed by the sensitive application developer.

The integrity and optionally the confidentiality of the application shall be maintained after the Off card verifier check or after the ITSEF evaluation and its loading on the TOE.

This check shall be ensured by the organisational measures or by security mechanisms.
The platform is evaluated without applications.

## 1.8 TOE Description

The TOE is composed of software and Hardware. The following chapters presents each part of the TOE.

### *1.8.1 Defensive Java Card Platform*

The Java technology, embedded on the TOE, combines a subset of the Java programming language with a runtime environment optimized for smart cards and similar small-memory embedded devices.

The Java Card™ platform is a smart card platform enabled with Java Card™ technology (also called, for short, a "Java Card"). This technology allows for multiple applications to run on a single card and provides facilities for secure interoperability of applications. Applications running on the Java Card platform ("Java Card applications") are called applets.

The TOE is compliant with the version of the Java Card 3.1 classic edition, specified by three documents related to Java Card API, Java Card Runtime Environment and Java Card Virtual Machine Specifications, defined respectively in **[JCAPI]**, **[JCRE]** and **[JCVM]** and . The next paragraph introduces those three elements.

As the terminology is sometimes confusing, the term "Java Card System" has been introduced in that defines the set constituted by the Java Card RE, the Java Card VM and the Java Card API.
The Java Card System provides an intermediate layer between the operating system of the card and the applications. This layer allows applications written for one smart card platform enabled with Java Card technology to run on any other such platform.

The Java Card VM is a bytecode interpreter embedded in the smart card. The Java Card RE is responsible for card resource management, communication, applet execution, on-card system and applet security.

Applet isolation is achieved through the Java Card Firewall mechanism defined in **[JCRE]**. This mechanism confines an applet to its own designated memory area. Thus, each applet is prevented from accessing fields and operations related to objects owned by other applets, unless those applets provide a specific interface (shareable interface) for that purpose. This access control policy is enforced at runtime by the Java Card VM.
However, applet isolation cannot be entirely granted by the firewall mechanism if certain well-formedness conditions are not satisfied by loaded applications.

Therefore, a bytecode verifier (BCV) formally verifies those conditions for a third party application. The BCV is out of the scope of the Java Card System defined in **[PP0099]**.

**The IDEMIA platform implements dynamic Verifier that allows the platform to be defensive. Verifications are done during execution of the byte code.**

And as this security target claims a demonstrable conformance to **[PP0099]**, the off card verifier is also used for third parties applications. In this case, applications are verified by the latest Oracle off card verifier.

IDEMIA can also provide a way for loading optimized Cap File converting the bytecode in a proprietary format; in this case, there is no need for these applications to go through off card byte code verification before the loading on the TOE. In this case, the applet certification ensures a consistent verification.

The Java Card API (JCAPI) provides classes and interfaces for the core functionality of a Java Card application. It defines the calling conventions by which an applet may access the JCRE and services

such as, among others, I/O management functions, PIN and cryptographic specific management and the exceptions mechanism. The JCAPI is compatible with formal international standards, such as ISO/IEC 7816 and industry specific standards.

Array Views allow creation of a "view" on a subset of elements of an existing array. The elements of the view are mapped to the elements of the actual array to avoid defensive copies, synchronization protocols, and to allow a fine-grained access control on the elements.

### *1.8.2 Global Platform*

The TOE is compliant with the Global Platform 2.3 (GP) standard **[GP1]** which provides a set of APIs and technologies to perform in a secure way, the operations involved in the management of the applications hosted by the card. Using GP maximizes the compatibility and the opportunities of communication as it becomes the current card management standard.
The main features addressed by GP are:

- The authentication of users through secure channels
- The downloading, installation removal, and selection for execution of Java Card applications
- The life cycle management of both the card and the applications
- The sharing of a global common PIN among all the applications installed on the card

These operations are addressed by a set of APIs used by the applications hosted on the card in order to communicate with the external world on a standard basis.
The version considered in this document is version 2.3 of the GP Card specification. The following GP functionalities, at least, are present within the TOE:

- Card content loading
- Extradition
- Asymmetric keys
- DAP support, Mandated DAP support
- DAP calculation with asymmetric cryptography
- Logical channels
- SCP02 support
- SCP03 support **[GP2]**
- Support for contact and contactless cards different implicit selection on different interfaces and channels
- Support for Supplementary Security Domains
- Trusted path privileges
- Post-issuance personalisation of Security Domain **[GP2]**
- Application personalisation **[GP2]**
- Crypto algorithms as detailed in 1.8.5.2 Cryptographic features

### *1.8.3 Integrated Circuit (IC), IFX Secure Smart Card Controller*

The platform is designed on IFX components:

#### 1.8.3.1 IFX Secure Smart Card Controller

The IC SLC37 is an IFX dual interface component that supports ISO/IEC 14443 Type A and type B.

It is a hardware device composed of a processing unit, memories, security components and I/O interfaces. It has to implement security features able to ensure:

- The confidentiality and the integrity of information processed and flowing through the device,

The resistance of the security IC to externals attacks such as physical tampering, environmental stress or any other attacks that could compromise the sensitive assets stored or flowing through it.

The IC configuration used in this project doesn't include any optional software or optional toolbox.

This IC is certified by the German BSI certification body. More information regarding the components is available in the public security target of the chips **[IFX_ST]** .

### *1.8.4 Crypto1*

The Crypto1 is a proprietary technology RFID dedicated to public transport, ticketing, access management based upon various levels of the ISO/IEC 14443 Type A 13.56 MHz contactless smart card standard.

It employs a proprietary protocol compliant to parts (but not all) of ISO/IEC 14443-3 Type A.

The Crypto1 does not offer any security function in the present evaluation.

### *1.8.5 Operating System (OS)*

The TOE relies on an Operating System (OS), which is an embedded piece of software loaded into the Security IC. The Operating System manages the features and resources provided by the underneath chip. It is, generally divided into two levels:

1) Low level:
   a) Drivers related to the I/O, RAM, SOLID Flash, and any other hardware component present on the Security IC
2) High level:
   a) Protocols and handlers to manage I/O
   b) Memory and file manager
   c) Cryptographic services and any other high level services provided by the OS

During its construction and its personalisation, the **FLEXICODE** feature allows deletion of the following modules (all or part of them):

- Crypto1 (out of TOE scope)
- Cipurse (out of TOE scope)
- DBI API
- RSA
- SHA3
- HMAC
- Cert (Certificate)
- CVCert (CV Certificate)
- Biometrics (MOC)
- PACE-CAM
- PACE-DH

By default the product does not include the listed modules.

#### 1.8.5.1 BIOS

The BIOS is an interface between hardware and native components like VM and APIs. The BIOS implements the following functionalities:

- APDU management, using T=0, T=1 and T=CL protocols (Type A and type B)
- Timer management

- Exceptions management
- Transaction management
- Flash memory access

### 1.8.5.2 Cryptographic features

The following crypto services are included in the OS:

| Cryptographic Services | Comments |
|---|---|
| RSA from 1024 to 4096-bits by step of 256-bits | References are standard ones |
| ECC with 192, 256, 384, 512 and 521-bits key sizes | |
| TDES with 56, 112 and 168-bits key sizes | |
| AES with 128, 192, 256 key sizes | |
| SHA-1, SHA 224, 256, 384 and 512, SHA-3 (for data integrity only does not provide confidentiality) | |
| RSA, ECC Key generation | |
| CRC 16 and CRC 32 (for data integrity only does not provide confidentiality) | |
| RNG CTR_DRBG SP800-90 | |
| RSA signature/verification | Based on supported RSA key sizes |
| ECDSA signature/verification | Based on supported ECC key sizes |
| ECDH | |
| AES secure messaging | References are standard ones |
| TDES secure messaging | |
| PIV secure messaging | |
| HMAC (64 bits up to 1016 bits) | |

### 1.8.5.3 Biometric feature

ID-One Cosmo X embeds the MOC algorithm.
The biometric feature allows matching a CANDIDATE Template with REFERENCE Templates (up to 10).
The MOC is in the scope of this ST for face recognition and fingerprint. The Iris feature is out of the scope of this ST.
Note: by default the product doesn't include this module.

### 1.8.5.4 Virtual Machine

The Virtual Machine, which is compliant with the Java Card 3.1 classic edition, interprets the byte code of Java Card applets.
The Virtual Machine supports logical channels; this means that it allows an applet to be selected on a channel, while a different applet is selected on another channel.
It also supports secure execution of applets loaded and stored in FLASH.
The Virtual Machine is activated upon the selection of an applet.

#### 1.8.5.5 The Java Card Runtime Environment

The Java Card Runtime Environment (JCRE) contains the Java Card Virtual Machine (VM), the Java Card Application Programming Interface (API) classes and industry-specific extensions, and support services. For details please refer to reference **[JCRE]**.

#### 1.8.5.6 APIs

The APIs, compliant with the Java Card 3.1 classic edition, support key generation, Key Agreement, signature, ciphering of messages and proprietary IDEMIA API.
Proprietary APIs **[AGD_PAPI]** have been developed utilBER_Reader to read BER-TLV or SecureStore to store integrity sensitive information.
Also, API for DBI are provided:
This DBI API is relative to the Digitally Blurred Image process (DBI). It allows the blurring of a JPG or JPEG2000 file stored in a transparent file. This module is part of the product and in the scope of this present certification.

The CVCert module manages the Card Verifiable Certificate (CVC) API with a DER encoded certificate. These certificates are generally dedicated to ID applets.
The Cert module manages the Certificate API generally dedicated to transportation applets.

The API Extensibility removes the CAP file limitation that prevented adding methods to the non-final classes. Additional information is added to the CAP file structure for an enhanced virtual method token assignment, which allows evolution of the platform API or shared libraries without breaking binary compatibility rules.

The Monotonic Counter API provides methods for creating a counter that can only be increased, never decreased.

The SensitiveArrays package provides methods for creating and handling integrity-sensitive array objects.

The Biometrics API provides the functionality to securely manage biometric templates, this includes:

- Atomic update of biometric reference templates and try counter,
- No rollback on the biometric-checking function,
- Keeping the reference template secret, once initialized,
- Enhanced protection of biometric template's security attributes (state, try counter…) in confidentiality and integrity.

#### 1.8.5.7 Open and isolating Platform

This security target claims conformance to the Application Note 10 on Open and Isolating platform, issued by ANSSI and by JIL **[NOTE10]**.
An "open platform" can host new applications:

- Before its delivery to the end user (during phases 4, 5 or 6 of the traditional smartcard lifecycle). Such loadings are called "pre-issuance".
- After its delivery to the end user (phase 7). Such loadings are called "post-issuance".

An "isolating platform" is a platform that maintains the separation of the execution domains of all embedded applications on a platform, as of the platform itself. "Isolation" refers here to domain separation of applications as well as protection of application's data.

#### 1.8.5.8 Resident Application

It provides a native code application, with a basic main dispatcher, to receive the card commands and dispatch them to the application and module functions to implement the application commands.
It also deals with the Card Manufacturer authentication.

Commands for administration are only available during pre-personalisation phase. After this phase, Card Manager and selected applet dispatchers receive the commands.

#### 1.8.5.9 Applets

Basic applets bytecodes shall go thru the latest Oracle off card verifier before the loading.
Sensitive proprietary applet including bytecodes that encodes the element to access in a more compact format shall be evaluated and certified (i.e. without going through Oracle off card verifier).

#### 1.8.5.10 JBOX

The JBox aims to embed a Third Party Library (TPL) that can be triggered using proprietary Java Card API. These elements are described as follows:

- Applets invoke TPL via a proprietary Java Card JBox API. It is a proprietary package. Notice that **[PP0099]** states that there is no difference between native (TPL in our case) and interpreted methods in their interface or invocation mechanism. Consequently, all the SFRs related to the Java Card API remains useful for the Java Card JBox API.
- TPL is native software, which is delivered by third party. It is flashed/loaded as a java card package in card during the development phase (manufacturing, pre-personalization or personalization) or during the usage phase thanks to the GlobalPlatform features with DAP. TPL can be loaded through ISD or SSD (with mandatory DAP). Loading and verification method may vary depending on phases and environment. Once loaded the package is linked. The link is composed of one entry point only that is defined at a fixed address. Notice that TPL calls shared memory to access to the data exchanged with the platform.
- The JBox is a native layer that manages the security around the execution of the TPL; i.e. it checks all the data exchanges between the Operating System and the TPL and it configures the hardware and software to make the TPL running safety for the platform. JBox uses the IC security functions: at least the MMU to switch context when executing the native code. Consequently, its implementation shall follow the IC security guidelines. The data are exchanged between through a shared memory located in Memory Manager.
- The context manages the execution environment of a piece of code (NVM, RAM, and resource accesses, etc.). Two contexts are defined: Platform Context and TPL Context. Notice that the JBox is the bridge between the platform context and the TPL context.
- Platform Context is dedicated to the platform execution. It must not access: (i) the TPL code; (ii) the TPL data except through the shared memory.
- TPL Context is dedicated to the TPL execution and the part of the JBox responsible of execution of the TPL. It must not access: (i) the platform code, except through the authorized Native services (the available operations, etc.); (ii) the platform data, except through shared memory.

It should be noticed that from a Java Card system point a view, there is no difference between native code (TPL in our case) and interpreted methods in their interface or invocation mechanism **[PP0099]**. Hence, a Java Card exception (specific or standard) will be triggered in case of runtime errors (no JBox configured on the product, empty TPL, etc.)

As the native code is converted to a Java Card code before its loading, the loading process of native code is the classical one used for Java Card applications loading and uses the same TOE security mechanisms. The new native code execution is made on its own context. If the issuer allows the native applications to be loaded, this operation occurs in any phase of the product life cycle indicated in section 1.11.

### 1.8.5.11 JPatch and CodOp

The platform allows to load patches at pre-personalisation, personalisation or use phase. The patches installed cannot be bypassed. The TOE identification is updated to take into account the patches installed.
The patches for applets (CodOp) are Any Java code in applet packages or Library packages stored in NVM.
The loading of any patch shall follow the procedure of impact analysis defined in the SOGIS Process Assurance Continuity.
If the patch reconsiders the security of the TOE, a reassessment of the TOE is mandatory, otherwise a maintenance process is used.

### 1.8.5.12 Cipurse

The CIPURSE open security standard was established by the Open Standard for Public Transportation (OSPT) Alliance to address the needs of local and regional transit authorities for automatic fare collection systems based on smart card technologies and advanced security measures.

The CIPURSE API are developed by IDEMIA. The "CIPURSE Server Crypto" API Rev 2.0 are included.

The Cipurse does not offer any security function in the present evaluation.

Note: by default the product doesn't include this module.

### 1.8.5.13 Password Authenticated Connection Establishment (PACE)

This package defines the PACE protocol according to ICAO Technical Report "Supplemental Access Control" **[ICAO-9303]** part 11 and conform to BSI-CC-PP-0068-V2 **[PP0068]** for protection of the communication between terminal and chip. The passport specificities can be indicated in application notes sections but the TSF keeps only the communication protocol. The PACE protocol can be used by an applet for an electronic passport or another application.

PACE is an access control mechanism that is supplemental to BAC (Basic access control). It is a cryptographically stronger access control mechanism since it uses asymmetric cryptography compared to BAC's symmetric cryptography.

Once the mutual authentication is performed, a secure messaging is available to protect the communication between the chip and the terminal.

The TOE proposes to activate the PACE for all Logical Channels and define in that way the Global Privacy Protocol (GPP) for PACE. The applets do not manage Secure Messaging and use the I/O "in clear" using standard APDU class, the secure messaging is automatically performed: the TOE automatically unwraps and wraps APDUs.

The TOE proposes also to activate the PACE on only one Logical Channel and define in that way the Local Privacy (LPP) for PACE. The applet can manages its secure messaging with wrap and unwrap APIs or use Automatic Secure Messaging (like the one in GPP).

The PACE protocol can be personalized during pre-personalization phase or personalization phase. Then the GPP is activated with this configured PACE.

The GPP and LPP are defined in the sense of GP Privacy Framework Requirements **[GP3]**.

The TOE supports the following protocols based on ECDH or DH:

- PACE-ECDH-GM (Generic mapping)
- PACE-ECDH-IM (Integrated mapping)
- PACE-ECDH-CAM (Chip Authentication), in option for CAM,
- PACE-DH-GM

And the following secure messaging associated to each protocol:

- 3DES-CBC-CBC (16 bytes keys, SHA-1)
- AES-CBC-CMAC-128 (16 bytes key, SHA-1)
- AES-CBC-CMAC-192 (24 bytes key, SHA-256)
- AES-CBC-CMAC-256 (32 bytes key, SHA-256)

- **PACE-DH**

The chip authentication with DH can be selected by the issuer of the product. The protocol PACE-DH is then used as specified in **[ICAO-9303]**. By default the product doesn't include PACE-DH.

- **PACE-CAM**

  The Chip Authentication Mapping is a new mapping for PACE which extends the Generic Mapping that integrates Chip Authentication into the PACE protocol. This mapping combines PACE and Chip Authentication into one protocol PACE-CAM, which allows faster execution than the separate protocols (i.e. PACE + Chip Authentication + Terminal Authentication).
  The chip computes the Chip Authentication Data using the chip's static private key then sends this data to the terminal. The terminal verifies the authenticity of the chip using the recovered Chip Authentication Data. As an additional PACE mode, Chip Authentication Mapping (PACE-CAM) defined in **[ICAO-9303]** part 11, which combines PACE-GM with Chip Authentication into a single protocol is optionally configured.
  By default the product doesn't include PACE-CAM.

## 1.9 Major Security feature of the TOE

The main goal of the TOE is to provide a sound and secure execution environment to critical assets that need to be protected against unauthorized disclosure and/or modification.
The TOE with its security function has to protect itself and protect applets from bypassing, abuse or tampering of its services that could compromise the security of all sensitive data. Even if the applets are not in the scope of this evaluation.

**Atomic Transactions**
The TOE shall provide a transaction mechanism. It shall execute a sequence of modifications and allocations on the persistent memory so that either all of them are completed, or the TOE behaves as if none of them had been attempted.
The transaction mechanism shall permit to update internal TSF data as well as to perform different functions of the TOE, like installing a new package on the card.
This mechanism shall be available for applet instances
The TOE shall perform the necessary actions to roll back to a safe state upon interruption.

**Card Content Management**
The TOE shall control the loading, installation, and deletion of packages and applet instances.
To remove the code of a package from the card, or to definitely deactivate an applet instance, so that it becomes no longer selectable; it shall perform physical removal of those packages and applet data stored in memories (except applet including in OS package in Flash memory that shall only be logically removed).

**Card Management Environment**
This function shall initialize and manage the internal data structure of the Card Manager. During the initialization phase of the card, it creates the Installer and the Applet Deletion Manager and initializes their internal data structures. The internal data structure of the Card Manager includes the Package and

Applet Registries, which respectively contains the currently loaded packages and the currently installed applet instances, together with their associated AIDs.
This function shall also be in charge of dispatching the APDU commands to the applet instances installed on the card and keeping trace of the currently active ones.
It therefore handles sensitive TSF data of other security functions, like the Firewall.

**Cardholder Verification**
The TOE shall implement mechanisms to identify and authenticate the user of the product. This function is available to applet instances.

**Clearing of sensitive information**
The TOE shall ensure that no residual information is available from memories, and shall protect sensitive information that is no longer used. The Platform has to securely clear and destroy this information. It concerns PINs, keys, sensitive data (such as D.BIO) and buffer APDU.
This function is also available to applet.

**DAP Verification**
An Application Provider may require that its Application code to be loaded on the card shall be checked for integrity and authenticity. The DAP Verification privilege of the Application Provider's Security Domain shall provide this service on behalf of the Application Provider. A Controlling Authority may require that all Application code to be loaded onto the card shall be checked for integrity and authenticity. The Mandated DAP Verification privilege of the Controlling Authority's Security Domain shall provide this service on behalf of the Controlling Authority.

**Data coherency**
As coherency of data should be maintained, and as power is provided by the CAD and might be stopped at all moment (by tearing or attacks), a transaction mechanism need to be implemented.
When updating data, before writing the new ones, the old ones are saved in a specific memory area. If a failure appears, at the next start-up, if old data are valid in the transaction area, the system restores them for staying in a coherent state.

**Data integrity**
Sensitive data have to be protected from modifications: keys, pins, patch code and sensitive applet data.

**Encryption and Decryption**
The TOE provides the applet instances with a mechanism for encrypting and decrypting the contents of a byte array.
Ciphering operations are implemented to resist environmental stress and glitches and include measures for preventing information leakage through covert channels.

**Entity authentication/secure Channel**
Off-card entity authentication is achieved through the process of initiating a Secure Channel and provides assurance to the card that it is communicating with an authenticated off-card entity.
If any step in the off-card authentication process fails, the process shall be restarted (i.e. new session keys generated).
The Secure Channel initiation and off-card entity authentication implies the creation of session keys derived from card static key(s).
In that way, the TOE provides Services to manage Global Platform authentication and secure messaging.

**Secure Messaging (SM)**
The SM provides security functionalities to encrypt/decrypt data for secure communication in Manufacturing, Personalization and Operational Use phases. A Secure Messaging session begins after a successful authentication (GP authentication for Pre-personalization and Personalization phases or Chip Authentication for instance in Operational Use phase). The Secure Messaging can be used in an

automatic mode: the enciphering and deciphering is automatically managed by the TOE. Whereas in mode "sequential" applet can still rely on a wrap / unwrap API.

**Exception**
In case of abnormal event: data unavailable on an allocation or illegal access to a data, the system shall own an internal mechanism allowing it to stop the code execution and raise an exception.

**Firewall**
The TOE with the Firewall shall control information flow at runtime. It shall ensure controls object sharing between different applet instances, and between applet instances and the Java Card RE.

**GP_Dispatcher**
While a Security Domain or Card Manager is selected, the TOE shall test for every command if Security Domain Owner authentication is required. If a secure channel is opened, the TOE tests according to the Security Domain state and the Card state for every command if secure messaging is required.

**Hardware operating**
The TOE shall boot after the IC has successfully powered-up. The TOE boot operations shall ensure the correct initialization of the TOE functionalities and the integrity of the code and data.
The TOE shall monitor IC detectors (e.g. out-of-range voltage, temperature, frequency, active shield, memory aging) and shall provide automatic answers to potential security violations through interruption routines that leave the device in a secure state.

**Key Access**
The TOE shall enforce secure access to all cryptographic keys on the card: RSA keys, DES keys, EC keys, AES keys

**Key Agreement**
The TOE shall provide to applet instances a mechanism for supporting key agreement algorithms such as EC Diffie-Hellman.

**Key destruction**
The TOE shall provide secure key destruction, such as keys cannot be retrieved from erased data.

**Key Distribution**
The TOE shall enforce the distribution of all the cryptographic keys of the card using a specific method.

**Key Generation**
The TOE shall enforce the creation and the on card generation of all the cryptographic keys of the card using a specific method.

**Key management**
The TOE shall manage key set: Loading keys, adding a new key set (version and value of the key) or updating a key set (update key value).

**Manufacturer Authentication**
During prepersonalisation phase, manufacturer authentication at the beginning of a communication session shall be mandatory prior to any relevant data being transferred to the TOE.

**Memory failure**
This security functionality is in charge of the management of bad usage of the memory.

**Message Digest**

The TOE shall provide the applet instances with a mechanism for generating an (almost) unique value for the contents of a byte array. This value can be used as a short representative of the information contained in the whole byte array.
For Hashing algorithms that do not pad the messages, the TSF checks that the information is block aligned before computing its hash value.

**Pre-personalisation and Patching**
This function shall permit to pre-initialize the internal data structures, to load the configuration of the card and to load patch code (with JPatch for the TOE or optional code loading for the applet) if needed in pre-personalization.
The TOE shall allow loading of TOE sensitive data: configuration data. Configuration data can contain patches. The TOE shall check the integrity of the incoming data. Unless stated otherwise, the origin of the incoming data shall be ensured by organisational means. The TOE shall ensure that TOE code and patches installed after delivery cannot be bypassed. The TOE identification shall take into account the patches installed after delivery.
CodOp (patch only for applets) lifecycle is equivalent to regular java CAP files. They are deployed like regular CAP file, after Security Domain authentication. CodOps deletion and replacement is possible.
As CAP file, CodOp deployment relies on Global Platform Security Architecture only after authentication to a CM enabled SD: ISD, Authorized or Delegated Management privileged SSD, with a possible DAP or Mandated DAP.

**JPatch at use phase and CodOp for applets**
The loading functionality of patches is also available in use phase, once installed the TOE identification shall take into account the patches installed after delivery.

**JBox**
This TOE functionality provides the user of the product some defined interfaces to create its own native application to load at on the platform. The application created is confined in its own execution environment and in its dedicated NVM and RAM. The JBox functionalities can be deactivated at any time of the product lifecycle.

**Random Number**
This TOE functionality provides the card manager, the resident application and the applets a mechanism for generating challenges and key values.
The Number Generator is a combination of hardware and software RNG. The RNG is compliant with **[NIST_RNG]**.

**Resident Application dispatcher**
During prepersonalisation phase, this function shall verify for every command if manufacturer authentication is required.

**Runtime Verifier**
This security functionality ensures the secure processing of the stack, heap and transient by ensuring additional controls.

**Security functions of the IC**
This TOE functionality ensures the correct execution of the IC functionalities.

**Signature**
This TSF shall provide the applet instances with a mechanism for generating an electronic signature of the contents of a byte array and verifying an electronic signature contained in a byte array.
An electronic signature is made of a hash value of the information to be signed, encrypted with a secret key. The verification of the electronic signature includes decrypting the hash value and checking that it actually corresponds to the block of signed bytes. Signature operations shall be implemented to resist

environmental stress and glitches and include measures for preventing information leakage through covert channels.

**Unobservability**
The TOE shall use and manipulate sensitive information without revealing any element of this information.

**CRC 16**
The TOE provides this security function to guarantee the integrity of the sensitive objects (such as keys or PINs) store in the card.

**PACE**
The TOE provides the PACE protocole on the Logical Channels with the possibility of an automatic secure messaging:

- All the type of PACE, including PACE-CAM;
- The chip authentication feature of PACE-CAM; note that PACE-CAM may be removed (code deletion) from the TOE during its configuration or with a lock during personalization.

**FLEXICODE**
The TOE provides a configuration command to remove parts of the code during its construction. The agent (personalizer or pre- personalizer) can securely delete pre-defined functionalities called Modules. The TOE provides the mean to identify list of modules that can be deleted. The FLEXICODE mechanism securely brings a good way to free memory flash available, for instance, to applets or user data with

- Authentication to access to deletion command and authorized modules,
- Preserving TOE implementation integrity with module independence,
- Maintaining the indicators for the module presence.

## 1.10 NON-TOE HW/SW/FW AVAILABLE TO THE TOE

The only non-TOE component required on the product is the bytecode verifier. The bytecode verifier is a program that performs static checks on the bytecodes of the methods of a CAP file.
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. Even if a dynamic verifier is implemented in the product, this TOE considers also static bytecode verification; it has to be performed on the host at off-card verification and prior to the loading of the file on the card in any case.
An exception is for verifying and checking IDEMIA bytecode, by the security evaluation laboratory. The off-card verifier is no longer required.

## 1.11 LIFE-CYCLE

The following description (see Tables 2 and 3) introduces generics but fine-grained 3 options for the life-cycle of secure smartcard products. These 3 options are compliant to standard smartcard life-cycle as defined in **[PP0099]** and **[PP0084]**. Since applets loading is outside the TOE, this document focuses on the Java Card platform (the TOE) life cycle which is part of the smart card product life cycle. The intent of the more fine-grained options is to cover the specific aspects of new technologies like platform loading in a comprehensive way and to add some flexibility with respect to the separation of responsibilities between the various parties involved. The smartcard product life-cycle is decomposed in 7 phases that describe the competent authorities for each of these phases.

| Phase | Phase name | Actors | Covered by |
|---|---|---|---|
| 1 | Security IC Embedded Software development | IDEMIA R&D<br>(Courbevoie, Noida-D and Pessac) | ALC[PLT] |
| 2 | Security IC Development | Infineon | ALC[IC] |
| 3 | Security IC Manufacturing<br>+ Platform Loading (in case of option a) ) | Infineon | ALC[IC] |
| | | | TOE Delivery |
| 4 | Security IC Packaging | Infineon or another agent | - |
| 5 | Platform Loading<br>(in case of option b) using IC Package 2 ) | IDEMIA plant<br>(Haarlem, Vitre, Ostrava, Shenzen and Noida-P)<br>or another agent | AGD_PRE |
| 6 | Pre-personalisation & Personalisation | IDEMIA plant<br>(Haarlem, Vitre, Ostrava, Shenzen and Noida-P)<br>or another agent | AGD_PRE |
| 7 | Operational Usage | The end user | AGD_OPE |

**Table 2: TOE Life (options a and b)**

| Phase | Phase name | Actors | Covered by |
|---|---|---|---|
| 1 | Security IC Embedded Software development | IDEMIA R&D<br>(Courbevoie, Noida-D and Pessac) | ALC[PLT] |
| 2 | Security IC Development | Infineon | ALC[IC] |
| 3 | Security IC Manufacturing | Infineon | ALC[IC] |
| 4 | Security IC Packaging | Infineon or another agent | - |
| 5 | Platform Loading<br>(in case of option c) using IC Package 1 ) | IDEMIA plant<br>(Haarlem, Vitre, Ostrava, Shenzen and Noida-P) | ALC[PLT] |
| | | | TOE Delivery |
| 6 | Pre-personalisation & Personalisation | IDEMIA plant<br>(Haarlem, Vitre, Ostrava, Shenzen and Noida-P)<br>or another agent | AGD_PRE |
| 7 | Operational Usage | The end user | AGD_OPE |

**Table 3: TOE Life (option c)**

*Notes:*

Phase

The applets and patches (with Jbox or Jpatch) if any can be loaded
- from phase 3 to phase 7 option a)
- from phase 5 to phase 7 option b) and c)

ALC[PLT] refers to IDEMIA audited sites
ALC[IC] refers to the IFX audited sites

Notice that the IC loader shall be blocked during the pre-personalisation and personalization phase; i.e. before the end-user delivery.

The loading in use phase, if performed, is done in accordance with **[NOTE6].**

### *1.11.1 Phase 1: Security IC Embedded Software development*

The platform Development is performed during Phase 1. This includes Java Card System (JCS) conception, design, implementation, testing and documentation. The development fulfilled requirements of the final product, including conformance to Java Card Specifications, and recommendations of the user guidance. The development is made in a controlled environment that avoids disclosure of source code, data and any critical documentation and that guarantees the integrity of these elements. The evaluation of the TOE includes the platform development environment.
The code and the associated data are sent

- To the IC manufacturer for loading on the IC, option a).
- To IDEMIA or third party sites, option b).
- To IDEMIA audited sites, option c).

### *1.11.2 Phase 2: Security IC Development*

The Composite Product life cycle covers Security IC development which is described in the IC ST identification (see **[IFX_ST]**).

### *1.11.3 Phase 3 and phase 4: Security IC Manufacturing and packaging*

The Phase 3 of the Composite Product life cycle covers the IC production and when required for option a) the loading of the platform code on the flash memory. This loading is done thanks to the IC security functions.
For options a) and b), the TOE delivery is at the end of phase 3 at any form factor of the chip (on wafer, modules, inlay, cards PVC/PETF or on die…).

Delivery method:

- ✓ For options a) the OS is delivered in HEX file encrypted with PGP key of the fab/chip manufacturer
- ✓ For options b)
  - The OS, with MSK/LSK already injected is encrypted with PGP of the third party,
  - Or the delivery is using IDEMIA Tools for Code delivery to IDEMIA audited sites.

### *1.11.4 Phase 5: Composite Product Integration*

Where the IC is directly delivered without the OS, the loading takes place at this phase. Two options are covered in this phase:

- Option b) At IDEMIA or third party sites (non-audited sites), only package 2 of the IC is available. The loading is done after mutual authentication, only authorised users are able to load the Platform Code.
- Option c) At only IDEMIA audited sites, the loading is done thanks to package 1 of the IC.

In case of option c, the TOE delivery is done at this step.

Delivery method:

- ✓ For options b) or c)
  - The OS, with MSK/LSK already injected is encrypted with PGP of the third party,
  - Or the delivery is using IDEMIA Tools for Code delivery to IDEMIA audited sites.

### 1.11.5 Phase 6: Composite Product Personalization

See preparative guidance.

### 1.11.6 Phase 7: Operational Usage

See operational guidance.

# 2 CONFORMANCE CLAIM

## 2.1 Common Criteria Conformance Claim

This security Target claims conformance to the Common Criteria version 3.1 revision 5, with the following documents:

| Common Criteria | Conformance rationale |
|---|---|
| Part 1 **[CC1]** | Conformant |
| Part 2 **[CC2]** | Extended with<br>The extended Security Functional Requirements are defined in chapter 6.1. |
| Part 3 **[CC3]** | Conformant |

**Table 4: CC conformance rationale**

This ST is compliant to assurance level EAL6 +, augmented with

- ALC_FLR.1: "Flow remediation".

## 2.2 Protection Profile Claim

This security target claims a demonstrable conformance to: **[PP0099]**
This security target is a composite security target, including the IC security target.
However, the security problem definition, the objectives, and the SFR of the IC are not described in this document.
As the IC is included in the TOE, OE.CARD-MANAGEMENT, OE.SCP.RECOVERY, OE.SCP.SUPPORT, and OE.SCP.IC are changed into the following Objectives on the TOE: O.CARD-MANAGEMENT, O.SCP.RECOVERY, O.SCP.SUPPORT, and O.SCP.IC.
As the SCP is included in the TOE, OE.NATIVE, OE.SCP.RECOVERY, OE.SCP.SUPPORT, and OE.SCP.IC are changed into the following Objectives on the TOE: O.NATIVE, O.SCP.RECOVERY, O.SCP.SUPPORT, and O.SCP.IC.
There are extra TOE objectives to provide additional services to applications. However, such extension has no impact on PP coverage.
There are extra Threats, OSP, Assumptions, TOE objectives and SFR without conflict with **[PP0099]**.
The RMI is not supported by the TOE, all related Threats, OSP, Assumptions, objectives and SFRs are then removed.
Finally as no other modification was done, we can conclude that the conformance is demonstrated
The product is in conformance with the minimum assurance level EAL6+ augmented with ALC_FLR.1.

## 2.3 Conformance rationale

This paragraph presents the consistency between the security target and the Java Card System Open configuration profile Protection Profile.

### *2.3.1 TOE SAR conformance*

The protection profile requires an assurance level of level EAL4 augmented with AVA_VAN.5 and ALC_DVS.2. This security target considers an assurance level EAL6 augmented with ALC_FLR.1, which still complies with the requirements of the protection profiles.

### 2.3.2 TOE Type conformance

The TOE type is in conformance with the TOE type described in the protection profile.
All SPDs of the PP [5] are included in this TOE. Justification for the conformance is done in following chapters.

### 2.3.3 SPD Statement Consistency

#### 2.3.3.1 Assets

All assets from the protection profile are included in the security target. Other assets have been added (see section 4.1.3).

#### 2.3.3.2 Threats

All threats from the protection profile are included in the security target. Other threats have been added (see section 4.3.9).

#### 2.3.3.3 OSPs

All OSPs from the protection profile are included in the security target.
Other OSPs have been added (see section 4.4.2).

#### 2.3.3.4 Assumptions

All the assumptions from the protection profile have been added in the security target, except A.DELETION.
A.DELETION has been removed from the security target because the deletion of applets is in the scope of the evaluation, as O.CARD_MANAGEMENT is an objective in this security target. Other assumptions have been added (see section 4.5.1).

#### 2.3.3.5 Objectives

#### 2.3.3.6 Security Objectives for the TOE

All the security objectives for the TOE from the protection profile are included in the security target.
Other security objectives for the TOE have been added (see section 5.1.6).

#### 2.3.3.7 Security Objectives for the Operational Environment

Not all the security objectives for the operational environment from the protection profile are included in the security target. The OE.APPLET is excluded with justification: Indeed: Instead of the OE.APPLET, this ST adds objectives O.JBox_FW on the TOE. In that way, applet with the native code is controlled by the platform with O.JBox_FW.

The loading and execution of native code is secured so as to not provide ways to bypass the TSFs of the JCS. These operations are submitted to the access contol and are part of the present evaluation.

These operations are done thru an access controlled by the MPU/MMU, the javacard Firewall, the JBOX Firewall and the OS. In that way, native code embedded in applet is under TOE security control including during post-issuance loading.

Some security objectives for the operational environment has been transformed in security objectives for the TOE, the rationale is presented in section section 2.2. Other security objectives for the operational environment have been added (see section 5.2.1).

#### 2.3.3.8 Security Functional Requirements

All SFRs from the protection profile have been added in the security target. Other SFRs have been added.to cover supplemental features:

- § 7.1.5.2 : SFR added for Card Manager,
- § 7.1.5.3 : SFR added for Resident Application,
- § 7.1.5.4 : SFR added for SmartCard Platform,
- § 7.1.5.5 : SFR added for the applets,
- § 7.1.5.6 : SFR added for Runtime Verification

- § 7.1.6 : SFR added for JBox.
- § 7.1.7 : SFR added for FLEXICODE.
- § 7.1.8 : SFR added for Monotonic counter.
- § 7.1.9 : SFR added for Sensitive Array.
- § 7.1.10 : SFR added for PACE.

- § 7.1.11 : SFR added for PACE-CAM module.
- § 7.1.12 : SFR added for RSA module.
- § 7.1.13 : SFR added for SHA-3 module.
- § 7.1.14 : SFR added for MOC module.
- § 7.1.15 : SFR added for PACE-DH module.
- § 7.1.16 : SFR added for HMAC module.

# 3 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.
For instance, we will define hereafter the following aspect:
#.OPERATE (1) The TOE must ensure continued correct operation of its security functions. (2) The TOE must also return to a well-defined valid state before a service request in case of failure during its operation.
TSFs must be continuously active in one way or another; this is called "OPERATE". The Security Target may include an assumption, called "A.OPERATE", stating that it is assumed that the TOE ensures continued correct operation of its security functions, and so on. However, it may also include a threat, called "T.OPERATE", to be interpreted as the negation of the statement #.OPERATE. In this example, this amounts to stating that an attacker may try to circumvent some specific TSF by temporarily shutting it down. The use of "OPERATE" is intended to ease the understanding of this document.
This section presents security aspects that will be used in the remainder of this document. Some being quite general, we give further details, which are numbered for easier cross-reference within the document. For instance, the two parts of #.OPERATE, when instantiated with an objective "O.OPERATE", may be met by separate SFRs in the rationale. The numbering then adds further details on the relationship between the objective and those SFRs.

## 3.1 Confidentiality

**#.CONFID-APPLI-DATA:**
Application data must be protected against unauthorized disclosure. This concerns logical attacks at runtime in order to gain read access to other application's data.

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

**#.CONFID-JCS-DATA:**
Java Card System data must be protected against unauthorized disclosure. This concerns logical attacks at runtime in order to gain a read access to Java Card System data. Java Card System data includes the data managed by the Java Card RE, the Java Card VM and the internal data of Java Card platform API classes as well.

## 3.2 Integrity

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

**#.INTEG-APPLI-DATA:**

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

**#.INTEG-APPLI-DATA-PHYS**
Integrity-sensitive application data must be protected against unauthorized modification by physical attacks.

**#.INTEG-JCS-CODE:**
Java Card System code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to executable code.

**#.INTEG-JCS-DATA:**
Java Card System data must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to Java Card System data. Java Card System data includes the data managed by the Java Card RE, the Java Card VM and the internal data of Java Card API classes as well.

## 3.3 Unauthorized executions

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

**#.EXE-JCS-CODE:**
Java Card System bytecode must be protected against unauthorized execution. Java Card System bytecode includes any code of the Java Card RE or API. This concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language([JAVASPEC], §6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code. Note that execute access to native code of the Java Card System and applications is the concern of #.NATIVE.

**#.FIREWALL:**
The Firewall shall ensure controlled sharing of class instances, and isolation of their data and code between packages (that is, controlled execution contexts) as well as between packages and the JCRE context. An applet shall not read, write, compare a piece of data belonging to an applet that is not in the same context, or execute one of the methods of an applet in another context without its authorization.

**#.JBox_FW:**
The JBox_FW shall ensure the isolation between the TSF of the JCS and the TPL. JBOX shall control the accces to the loaded native code (O.TPL_Content). It shall also ensure the isolation of TPL data and code and shall confine its access and its execution in the dedicated NVM and RAM.

**#.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.

The Native library (TPL) shall not bypass the TSFs of the JCS. The services available to the native code thru JBox shall be controlled and restricted to the OP.Native_Access operations.

The accesses shall be controlled by the MPU/MMU, the javacard Firewall and the OS access control.

## 3.4 Bytecode verification

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

### *3.4.1 CAP file verification*

#### 3.4.1.1 CAP file verification for standard bytecodes applications

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

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

- first, the assurance that every package required by the loaded applet is indeed on the card, in a binary-compatible version (binary compatibility is explained in **[JCVM]** §4.4),
- second, that the export files used to check and link the loaded applet have the corresponding correct counterpart on the card.

Also, introduced in version 3.1 of Java Card Specifications, Static Resources in CAP files, allow an application to embed static resources such as configuration data or initialization data, in the CAP file and access these resources from the application code.

#### 3.4.1.2 CAP file verification for proprietary bytecode applications

IDEMIA made some additions and optimizations to byte code instructions. These additions are part of the evaluation by the security lab and do not add any vulnerability on the firewall.

### 3.4.2 Integrity and authentication

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

### 3.4.3 Linking and authentication

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

## 3.5 Card management

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

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

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

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

**#.DELETION:**
(1) Deletion of installed applets (or packages) should not introduce security holes in the form of broken references to garbage collected code or data, nor should they alter integrity or confidentiality of

remaining applets. The deletion procedure should not be maliciously used to bypass the TSFs. (2) Erasure, if deemed successful, shall ensure that any data owned by the deleted applet is no longer accessible (shared objects shall either prevent deletion or be made inaccessible). A deleted applet cannot be selected or receive APDU commands. Package deletion shall make the code of the package no longer available for execution. (3) Power failure or other failures during the process shall be taken into account in the implementation so as to preserve the SFRs. This does not mandate, however, the process to be atomic. For instance, an interrupted deletion may result in the loss of user data, as long as it does not violate the SFRs.
The deletion procedure and its characteristics (whether deletion is either physical or logical, what happens if the deleted application was the default applet, the order to be observed on the deletion steps) are implementation-dependent. The only commitment is that deletion shall not jeopardize the TOE (or its assets) in case of failure (such as power shortage).
Deletion of a single applet instance and deletion of a whole package are functionally different operations and may obey different security rules. For instance, specific packages can be declared to be undeletable (for instance, the Java Card API packages), or the dependency between installed packages may forbid the deletion (like a package using super classes or super interfaces declared in another package).

**#.FLEXICODE**
(1) Deletion of modules (part of code) should not introduce security holes in the form of references pointing to memory zones that are no longer in use, or have been reused for other purposes. Deletion of modules should not be maliciously used to circumvent the TSFs. (2) Erasure, if deemed successful by atomic transaction, shall ensure that the deleted module is no longer accessible (3) integrity of TOE source code shall be preserved.

## 3.6 Services

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

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

**#.RESOURCES:**
The TOE controls the availability of resources for the applications 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) Protection of PIN's security attributes (state, try counter, try limit…) 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 **[PP0084]**), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

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

# 4 Security Problem Definition

## 4.1 Assets

Assets are security-relevant elements to be directly protected by the TOE. Confidentiality of assets is always intended with respect to un-trusted people or software, as various parties are involved during the first stages of the smart card product life-cycle; details are given in threats hereafter.

Assets may overlap, in the sense that distinct assets may refer (partially or wholly) to the same piece of information or data. For example, a piece of software may be either a piece of source code (one asset) or a piece of compiled code (another asset), and may exist in various formats at different stages of its development (digital supports, printed paper). This separation is motivated by the fact that a threat may concern one form at one stage, but be meaningless for another form at another stage.

The assets to be protected by the TOE are listed below. They are grouped according to whether it is data created by and for the user (User data) or data created by and for the TOE (TSF data). For each asset it is specified the kind of dangers that weigh on it.

### *4.1.1 User data*

**D.APP_CODE**

The code of the applets and libraries loaded on the card.

To be protected from unauthorized modification.

**D.APP_C_DATA**

Confidentiality - sensitive data of the applications, like the data contained in an object, an array view, 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, an array view,and the PIN security attributes (PIN Try limit, PIN Try counter and State).

To be protected from unauthorized modification.

*Application Note:*

The scope of D.APP_I_DATA is widened in view of the Monotonic counters functionality, i.e. the asset now also address and cover monotonic counters when the corresponding Module is embedded.

**D.APP_KEYs**

Cryptographic keys owned by the applets.

To be protected from unauthorized disclosure and modification.

**D.PIN**

Any end-user's PIN.

To be protected from unauthorized disclosure and modification.

### *4.1.2 TSF data*

**D.API_DATA**

Private data of the API, like the contents of its private fields.

To be protected from unauthorized disclosure and modification.

**D.CRYPTO**

Cryptographic data used in runtime cryptographic computations, like a seed used to generate a key.

To be protected from unauthorized disclosure and modification.

**D.JCS_CODE**

The code of the Java Card System.

To be protected from unauthorized disclosure and modification.

**D.JCS_DATA**

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

To be protected from unauthorized disclosure or modification.

**D.SEC_DATA**

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

To be protected from unauthorized disclosure and modification.

### *4.1.3 Additional assets*

**D.CONFIG**

The configuration DATA are put at prepersonalisation phase: locks, keys, patch if any. These elements of configuration have to be loaded securely. To be protected from unauthorized disclosure or modification.

**D.SENSITIVE_DATA**

The other sensitive data are grouped in the same D.Sensitive_Data. The list is presented below:

- D.NB_AUTHENTIC: Number of authentications. This number is specified in the SFR

- D.NB_REMAINTRYOWN: Number of remaining tries for owner PIN. This number is specified in the SFR
- D.NB_REMAINTRYGLB: Number of remaining tries for a global PIN. This number is specified in the SFR
- ASG.CARDREG: Card registry (AS.APID: Applet Identifier (AID), AS.CMID: Card Manager ID (AID))
- ASG.APPRIV: Applet privileges group (Card Manager lock privilege, Card terminate privilege, Default selected privilege, PIN change privilege, Security Domain privilege, Security Domain with DAP verification privilege, Security Domain with Mandated DAP verification privilege)
- AS.AUTH_MSK_STATUS: Authentication MSK Status, this Security Attribute verifies if the authentication with the MSK key is performed successfully or nO.PACE_
- AS.CMLIFECYC: this Security Attribute represents the Card life cycle state. It can be either: Prepersonalisation, Personalisation and use phases of the card.

**D.ARRAY**

Applets are enabled to store confidential data. To be protected from unauthorized disclosure and modification.

**D.JCS_KEYS**

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

*Application Note:*

In the scope of National Identity Card, for instance, a need to install ciphered applet after wafer delivery implied to encrypt the blocks for applet installation.A dedicated key is used. The ISD who has the privilege CLFBD can use the key automatically created from LSK. The SD provider can also create its onw keys under its SD with CLFBD.

**D.PERSO_DUMP**

Memory blocks in NVM representing Applet configuration structures to be extracted and written at prepersonalisation phase. To be protected from unauthorized disclosure or modification.

**D.CLFDB-DK**

Symmetric key to be used to decrypt Load File Data Blocks. To be protected from unauthorised disclosure and modification.

### *4.1.4 Assets for FLEXICODE package*

**D.MODULES**

D.Modules is modules, e.g. the independant parts of code that can be deleted during the TOE construction. The TOE is delivered and installed on IC, then the modules can be deleted dur pre-personnalization or personnalization.

### 4.1.5 Assets for PACE Package

#### 4.1.5.1 Primary assets

All these primary assets represent User Data in the sense of the CC.

**User data stored on the TOE**

All data (being not authentication data) stored in the context of the application(s) on the electronic document. These data are allowed to be accessed by an authenticated terminal acting as a PACE Terminal with appropriate authorization level. The generic security properties to be maintained by the current security policy are: Confidentiality, Integrity, Authenticity

*Application Note:*

Please note that user data being referred to in the table above include, amongst other, individual-related (personal) data of the TOE apllication holder (e.g. document) which also include his sensitive (i.e. biometric) data. Hence, the general security policy defined by the current ST also secures these specific holder's data as stated in the assets. This asset covers "User Data on the MRTD's chip", "Logical MRTD Data" and "Sensitive User Data" in [BAC-PP].

**User data transferred between the TOE and the terminal connected**

All data, with the exception of authentication data, that are transferred (both directions) during usage of the application(s) of the electronic document between the TOE and authenticated terminals. Generic Security Properties: Confidentiality, Integrity, Authenticity

#### 4.1.5.2 Secondary assets

The secondary assets also having to be protected by the TOE in order to achieve a sufficient protection of the primary assets are:

**Accessibility to the TOE functions and data only for authorised subjects**

Property of the TOE to restrict access to TSF and TSF-data stored in the TOE to authorized subjects only. The property to be maintained by the current security policy is: Availability

**TOE internal secret cryptographic keys**

Permanently or temporarily stored secret cryptographic material used by the TOE in order to enforce its security functionality. The properties to be maintained by the current security policy are: Confidentiality, Integrity

*Application Note:*

Application Note: Data for electronic document holder authentication and for authorization of communication with the electronic document can be categorized as (i) reference information that are persistently stored within the TOE, and (ii) verification information for the TOE that are input by a human user during an authentication and/or authorization attempt. The TOE shall secure both reference information, and, together with the connected terminal, verification information that are transferred in the channel between the TOE and the terminal.

**TOE internal non-secret cryptographic material**

Permanently or temporarily stored non-secret cryptographic (public) keys and other non-secret material (Document Security Object SOD containing digital signature) used by the TOE in order to enforce its security functionality. The properties to be maintained by the current security policy are: Integrity, Authenticity

**PACE Communication establishment authorisation data**

Restricted-revealable authorization information for a human user being used for verification of the authorisation attempts as authorized user (PACE password). These data are stored in the TOE and are not to be send to it. The properties to be maintained by the current security policy are: Confidentiality, Integrity

*Application Note:*

TOE application communication establishment authorisation data are represented by two different entities: (i) reference information being persistently stored in the TOE and (ii) verification information being provided as input for the TOE by a human user as an authorisation attempt.The TOE shall secure the reference information as well as – together with the terminal connected (the input device of the terminal) – the verification information in the 'TOE? terminal' channel, if it has to be transferred to the TOE. Please note that PACE passwords are not to be send to the TOE.

The secondary assets represent TSF and TSF-data in the sense of the CC.

### *4.1.6 Additional assets for Biometrics (MOC) - MODULE*

**D.BIO**

Any biometric template. To be protected from unauthorized disclosure and modification.

*Application Note:*

The asset is in the scope of this ST for face recognition and fingerprint. However, the Iris feature is out of the scope of this ST.

This asset is similar to D.PIN asset. The handling of D.BIO is performed in the same way than D.PIN. The same objectives, threats, SFR and others relevant security elements applicable to D.PIN are applicable to D.BIO.

## 4.2 Users / Subjects

### *4.2.1 Additional Users / Subjects*

**S.RESIDENT_APPLICATION**

The resident application

**R.personaliser**

Card Issuer or card Manufacturer

**R.Prepersonaliser**

Card manufacturer

**R.Card_Manager**

Card Manager

**R.Security_Domain**

Application Provider and its associated Security Domain

**R.Use_API**

User of applet for identification

**R.Applet_privilege**

Applet administrator

### *4.2.2 Miscellaneous*

**U.Card_Issuer**

The Card Issuer is the entity that own the card and is ultimately responsible for the behaviour of the card. It is initially the only entity authorized to manage applications through a secure communication channel with the card.

**U.Card_Manufacturer**

The Card Manufacturer is the entity responsible for producing smart cards on behalf of the Card Issuer.

**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.

**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 CAP files. This subject is involved in the CAP FILE LOADING security policy.

**S.CAD**

The CAD represents the actor that requests, by issuing commands to the card. 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**

Operands 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.CAP_FILE**

A CAP file may contain multiple Java language packages. A package is a namespace within the Java programming language that may contain classes and interfaces. A CAP file may contain packages that define either user library, or one or several applets. A CAP file compliant with Java Card Specifications version 3.1 may contain multiple Java language packages. An EXTENDED CAP file as specified in Java Card Specifications version 3.1 may contain only applet packages, only library packages or a combination of library packages. A COMPACT CAP file as specified in Java Card Specifications version 3.1 or CAP files compliant to previous versions of Java Card Specification, MUST contain only a single package representing a library or one or more applets.

**S.TOE**

Source code.

**S.TPL**

It is the native TPL in the JBox.

**S. Platform**

It is the Java Card platform without TPL.

### *4.2.3 Users / Subjects for PACE package*

**S.EDH Electronic document holder**

A person who the electronic document issuer has personalized the electronic document for. Personalization here refers to associating a person uniquely with a specific electronic document. Note that an electronic document holder can also be an attacker.

**S.EDP Electronic document presenter**

A person presenting the electronic document to a terminal and claiming the identity of the electronic document holder. Note that an electronic document presenter can also be an attacker, cf. below.

**S.Terminal**

A terminal is any technical system communicating with the TOE either through the contact interface or through the contactless interface. The role 'Terminal' is the default role for any terminal being recognised by the TOE as not being PACE authenticated ('Terminal' is used by the electronic document presenter).

*Application Note:*

This entity is commensurate with 'Terminal' in [BAC-PP].

**S.Terminal PACE (BIS-PACE): Basic Inspection System**

A technical system being used by an inspecting authority and verifying the electronic document presenter as the Electronic Document holder. BIS-PACE implements the terminal's part of the PACE protocol and authenticates itself to the electronic document using a shared password (PACE password) and supports Passive Authentication.

*Application Note:*

For ePassport: by comparing the real biometric data (face) of the travel document presenter with the stored biometric data (DG2) of the travel document holder.

**S.Personalisation Agent**

An organization acting on behalf of the electronic document Issuer to personalise the electronic document for the electronic document holder by some or all of the following activities: (i) establishing the identity of the electronic document holder for the biographic data in the electronic document, (ii) enrolling the biometric reference data of the electronic document holder, (iii) writing a subset of these data on the physical electronic document (optical personalisation) and storing them in the electronic document (electronic personalisation) for the electronic document holder, (iv) writing the document details data, (v) writing the initial TSF data, (vi) signing the Document Security Object and the elementary files EF.CardSecurity and the EF.ChipSecurity (if applicable [ICAO9303], **[TR_03110]**) (in the role of DS). Please note that the role 'Personalisation Agent' may be distributed among several institutions according to the operational policy of the electronic document Issuer.

**S.Manufacturer**

Generic term comprising both the IC manufacturer that produces the integrated circuit, and the electronic document manufacturer that creates the electronic document and attaches the IC to it. The manufacturer is the default user of the TOE during the manufacturing life cycle phase. When referring to the role manufacturer, the TOE itself does not distinguish between the IC manufacturer and the electronic document manufacturer.

**S.Attacker**

A threat agent (a person or a process acting on his behalf) trying to undermine the security policy, especially to change properties of the assets having to be maintained. The attacker is assumed to possess an at most high attack potential. Please note that the attacker might 'capture' any subject role recognised by the TOE.

*Application Note:*

For a travel document (if applicable [ICAO9303], **[TR_03110]**): Additionally to this definition, the definition of an attacker is refined as follows: A threat agent trying (i) to

manipulate the logical travel document without authorization, (ii) to read sensitive biometric reference data (i.e. EF.DG3, EF.DG4), (iii) to forge a genuine travel document, or (iv) to trace a travel document.

## 4.3 Threats

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

### *4.3.1 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 Java Card System code. See #.CONFID-JCS-CODE for details.

Directly threatened asset(s): D.JCS_CODE.

**T.CONFID-JCS-DATA**

The attacker executes an application to disclose data belonging to the Java Card System. See #.CONFID-JCS-DATA for details.

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

### *4.3.2 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 Java Card System code. See #.INTEG-JCS-CODE for details.

Directly threatened asset(s): D.JCS_CODE.

**T.INTEG-JCS-DATA**

The attacker executes an application to alter (part of) Java Card System or API data. See #.INTEG-JCS-DATA for details.

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

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

### *4.3.3 IDENTITY USURPATION*

**T.SID.1**

An applet impersonates another application, or even the Java Card RE, in order to gain illegal access to some resources of the card or with respect to the end user or the terminal. See #.SID for details.

Directly threatened asset(s): D.SEC_DATA (other assets may be jeopardized should this attack succeed, for instance, if the identity of the JCRE is usurped), D.PIN, D.BIO, D.JCS_KEYS and D.APP_KEYs and D.SENSITIVE_DATA.

**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) and D.SENSITIVE_DATA.

### 4.3.4 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.

*Application Note:*

This Threat is maintained and extended with the addition of T.JBox_BORDER.

### 4.3.5 DENIAL OF SERVICE

**T.RESOURCES**

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

Directly threatened asset(s): D.JCS_DATA.

### 4.3.6 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, D.APP_CODE and D.SENSITIVE_DATA.

**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) and D.SENSITIVE_DATA.

### 4.3.7 SERVICES

**T.OBJ-DELETION**

The attacker keeps a reference to a garbage collected object in order to force the TOE to execute an unavailable method, to make it to crash, or to gain access to a memory containing data that is now being used by another application. See #.OBJ-DELETION for further details.

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

### 4.3.8 MISCELLANEOUS

**T.PHYSICAL**

The attacker discloses or modifies the design of the TOE, its sensitive data or application code by physical (opposed to logical) tampering means. This threat includes IC failure analysis, electrical probing, unexpected tearing, and DPA. That also includes the modification of the runtime execution of Java Card System or SCP software through alteration of the intended execution order of (set of) instructions through physical tampering techniques.

This threatens all the identified assets in the present evaluation (restricted to physical attacks).

This threat refers to the point (7) of the security aspect #.SCP, and all aspects related to confidentiality and integrity of code and data.

*Application Note:*

The attacker performs a physical manipulation to alter (part of) an application's integrity-sensitive data. See #.INTEG-APPLI-DATA-PHYS for details. Directly threatened asset(s): D.APP_I_DATA, D.PIN, and D.APP_KEYs.

### 4.3.9 Additional threats

**T.CONFIGURATION**

The attacker tries to observe or modify configuration information exchanged between the TOE and its environnment. The TOE in this phase (prepersonalisation) must protect itself from modification or theft. Even the field is protected by assurance measures, each operations realised in this phase has to be protected.

Directly threatened asset(s): D.CONFIG, D.PERSO_DUMP

**T.CONF_DATA_APPLET**

The attacker tries to observe the operation of comparison between two byte arrays in order to catch confidential information manipulated.

Directly threatened asset(s): D.ARRAY

**T.PATCH_LOADING**

The attacker tries to avoid the loading of a genuine patch by:

- altering a patch (during loading or once loaded),

- exploiting the patch loading mechanism to load unauthenticated code on the TOE

in order to get access to the assets, the TSF data or the TOE user data, or to modify the TSF.

Directly threatened asset(s): D.CONFIG

**T.JBox_BORDER**

An attacker may try to load, modify and execute TPL in the JBox to modify the correct behaviour of the platform. With the aim to (i) disclose the OS code, (ii) disclose or alter applet code, disclose or alter OS data or disclose or alter applet data. Directly threatened asset(s): all.

**T.PERSO_DUMP**

The attacker tries to alter or observe the memory image extracted or written for applet personalization configuration by:

- altering a memory image (during loading or once loaded) creating a genuine applet personalization configuration,
- exploiting the dump extracting mechanism or dump memory blocks to read unauthenticated configuration on the TOE

in order to get access to the assets, the TSF data or the TOE user data, or to modify the TSF.

Directly threatened asset(s): D.PERSO_DUMP

**T.CLFDB-DISC**

The attacker discloses a Ciphered Load File Data Block when it is transmitted to the TOE for decryption prior to installation

Directly threatened asset(s): D.APP_CODE, D.JCS_KEY

*Application Note:*

This threat refines T.CONFID-JCS-DATA to address the CLFDB.

### *4.3.10 Additional threats for FLEXICODE package*

**T.FLEXICODE**

An attacker try to alter the TOE integrity to modify the correct behaviour of the platform by exploiting the module deletion

- to delete unexpected module or other parts of the code
- to get access to the assets, the TSF data or the TOE user data

Directly threatened asset(s): D.MODULES, the TOE itself and associated data.

### 4.3.11 Threats for PACE package

The Threats for PACE Module are refined from ICAO-PACE PP **[PP0068]** in a more generic form in order to be applicable to any application requiring PACE protocol. The eMRTD is kept as application exemple. The naming T.xxx from PP is transformed in T.PACE_xxxx in this ST.

**T.PACE_Skimming**

***Capturing Card-Terminal Communication*** Adverse action: An attacker imitates an inspection system in order to get access to the user data stored on or transferred between the TOE and the inspecting authority connected via the contactless/contact interface of the TOE. Threat agent: having high attack potential, cannot read and does not know the correct value of the shared password (PACE password) in advance. Asset: confidentiality of logical electronic document data.

**T.PACE_Eavesdropping**

***Eavesdropping on the communication between the TOE and the PACE terminal*** Adverse action: An attacker is listening to the communication between the electronic document and the PACE authenticated BIS-PACE in order to gain the user data transferred between the TOE and the terminal connected. Threat agent: having high attack potential, cannot read and does not know the correct value of the shared password (PACE password) in advance. Asset: confidentiality of logical electronic document data.

**T.PACE_Forgery**

***Forgery of Data*** Adverse action: An attacker fraudulently alters the User Data or/and TSF-data stored on the electronic document or/and exchanged between the TOE and the terminal connected in order to outsmart the PACE authenticated BIS-PACE or EIS-PACE by means of changed electronic document holder's related reference data (like biographic or biometric data). The attacker does it in such a way that the terminal connected perceives these modified data as authentic one. Threat agent: having high attack potential. Asset: integrity of the electronic document.

**T.PACE_Abuse-Func**

***Abuse of Functionality*** Adverse action: An attacker may use functions of the TOE which shall not be used in TOE operational phase in order (i) to manipulate or to disclose the User Data stored in the TOE, (ii) to manipulate or to disclose the TSF-data stored in the TOE or (iii) to manipulate (bypass, deactivate or modify) soft-coded security functionality of the TOE. This threat addresses the misuse of the functions for the initialisation and personalisation in the operational phase after delivery to the electronic document holder. Threat agent: having high attack potential, being in possession of one or more legitimate electronic documents. Asset: integrity and authenticity of the electronic document, availability of the functionality of the electronic document.

**T.PACE_Information_Leakage**

***Information Leakage from electronic document*** Adverse action: An attacker may exploit information leaking from the TOE during its usage in order to disclose confidential User Data or/and TSF-data stored on the electronic document or/and exchanged between the TOE and the terminal connected. The information leakage may be inherent in the

normal operation or caused by the attacker. Threat agent: having high attack potential. Asset: confidentiality of User Data and TSF-data of the electronic document.

**T.PACE_Phys-Tamper**

***Physical Tampering*** Adverse action: An attacker may perform physical probing of the electronic document in order (i) to disclose the TSF-data, or (ii) to disclose/reconstruct the TOE's Embedded Software. An attacker may physically modify the electronic document in order to alter (i) its security functionality (hardware and software part, as well), (ii) the User Data or the TSF-data stored on the electronic document. Threat agent: having high attack potential, being in possession of one or more legitimate electronic documents. Asset: integrity and authenticity of the electronic document, availability of the functionality of the electronic document, confidentiality of User Data and TSF-data of the electronic document.

**T.PACE_Malfunction**

***Malfunction due to Environmental Stress*** Adverse action: An attacker may cause a malfunction the electronic document's hardware and Embedded Software by applying environmental stress in order to (i) deactivate or modify security features or functionality of the TOE' hardware or to (ii) circumvent, deactivate or modify security functions of the TOE's Embedded Software. This may be achieved e.g. by operating the electronic document outside the normal operating conditions, exploiting errors in the electronic document's Embedded Software or misusing administrative functions. To exploit these vulnerabilities an attacker needs information about the functional operation. Threat agent: having high attack potential, being in possession of one or more legitimate electronic documents, having information about the functional operation Asset: integrity and authenticity of the electronic document, availability of the functionality of the electronic document, confidentiality of User Data and TSF-data of the electronic document.

## 4.4 Organisational Security Policies

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

### *4.4.1 OSP for Verification*

**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.

*Application Note:*

OE.VERIFICATION guarantees the correct integrity and authenticity evidences for each application, by means of elements provided by OE.CODE-EVIDENCE.

### 4.4.2 Organisational Security policies for PACE package

**P.PACE_Manufact**

***Manufacturing of the electronic document's chip*** The Initialisation Data are written by the IC Manufacturer to identify the IC uniquely. The electronic document Manufacturer writes the Pre-personalisation Data which contains at least the Personalisation Agent Key.

**P.PACE_Pre-operational**

***Pre-operational handling of the TOE and associated applications*** 1.) The electronic document Issuer issues the electronic document and approves it using the terminals complying with all applicable laws and regulations. 2.) The electronic document Issuer guarantees correctness of the user data (amongst other of those, concerning the electronic document holder) and of the TSF-data permanently stored in the TOE. 3.) The electronic document Issuer uses only such TOE's technical components (IC) which enable traceability of the electronic documents in their manufacturing and issuing life cycle phases, i.e. before they are in the operational phase. 4.) If the electronic document Issuer authorises a Personalisation Agent to personalise the electronic document for electronic document holders, the electronic document Issuer has to ensure that the Personalisation Agent acts in accordance with the electronic document Issuer's policy.

**P.PACE_Terminal**

***Abilities and trustworthiness of terminals*** The Basic Inspection Systems with PACE (BIS-PACE) shall operate their terminals as follows: 1.) The related terminals (basic inspection system, cf. above) shall be used by terminal operators and by electronic document holders. 2.) They shall implement the terminal parts of the PACE protocol, of the Passive Authentication and use them in this order. The PACE terminal shall use randomly and (almost) uniformly selected nonces, if required by the protocols (for generating ephemeral keys for Diffie-Hellmann). 3.) The related terminals need not to use any own credentials. 4.) They shall also store the Country Signing Public Key and the Document Signer Public Key (in form of CCSCA and CDS) in order to enable and to perform Passive Authentication (determination of the authenticity of data groups stored in the electronic document). 5.) The related terminals and their environment shall ensure confidentiality and integrity of respective data handled by them (e.g. confidentiality of PACE passwords, integrity of PKI certificates, etc.), where it is necessary for a secure operation of the TOE according to the current ST.

**P.PACE_Personalisation**

***Personalisation of the electronic document by issuing State or Organisation only*** The issuing State or Organisation guarantees the correctness of the biographical data, the printed portrait and the digitized portrait, the biometric reference data and other data of the logical electronic document with respect to the electronic document holder. The personalisation of the electronic document for the holder is performed by an agent authorized by the issuing State or Organisation only.

### 4.4.3 OSP for CLFDB

**OSP.CLFDB_ENC**

The Load File Data Block must be encrypted securely by a trusted SD provider.

*Application Note:*

See **[GP1]** section C.6.

## 4.5 Assumptions

This section introduces the assumptions made on the environment of the TOE. Due to the Protection Profile and Security Target definition, T.DELETION replaces A.DELETION as O.CARD_MANAGEMENT replaces OE.CARD_MANAGEMENT. Also T.JBox_BORDER remplaces A.CAP_FILE. Application Note: The Java Card Applets loaded pres and post-issuance and without JBox API do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM22], §3.3) outside the API.

### 4.5.1 Asumptions for Verification

**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. The platform allows the loading of IDEMIA proprietary bytecode for evaluated applets. Then the evaluation lab does the verifications.

*Application Note:*

The verifications concerns the javacard application. Native code are not concerned by this assumption. Applets loaded post-issuance and without JBox API do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM22], §3.3) outside the API.

**A.CAP_FILE**

CAP Files loaded post-issuance (without using JBox mechanism) do not contain native methods. The Java Card specification explicitly "does not include support for native methods" (**[JCVM]**, §3.3) outside the API.

### 4.5.2 Asumptions for PACE package

**A.PACE_Insp_Sys**

***Inspection Systems for global interoperability*** The Extended Inspection System (EIS) for global interoperability includes the Country Signing CA Public Key and implements the terminal part of PACE **[ICAO-9303]** part 11. If others protocoles are supported by the TOE and the IS, PACE must be used. The EIS reads the logical electronic document under PACE and establishes secure messaging. Justification: The assumption A.PACE_Insp_Sys does not confine the security objectives of the **[PP0068]** as it repeats the requirements of P.PACE_Terminal and adds only assumptions for the Inspection Systems.

# 5 Security Objectives

## 5.1 Security Objectives for the TOE

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

SFRs related to RMI functionality are excluded, as they are not part of this security target.

### *5.1.1 IDENTIFICATION*

**O.SID**

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

### *5.1.2 EXECUTION*

**O.FIREWALL**

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

**O.GLOBAL_ARRAYS_CONFID**

The TOE shall ensure that the APDU buffer that is shared by all applications is always cleared 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 cleared after the return from the install method.

**O.GLOBAL_ARRAYS_INTEG**

The TOE shall ensure that no application can store a reference to the APDU buffer, a global byte array created by the user through makeGlobalArray method and the byte array used for invocation of the install method of the selected applet.

**O.NATIVE**

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

**O.OPERATE**

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

**O.REALLOCATION**

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

**O.RESOURCES**

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

**O.ARRAY_VIEWS_CONFID**

The TOE shall ensure that no application can read elements of an array view not having array view security attribute ATTR_READABLE_VIEW. The TOE shall ensure that an application can only read the elements of the array view within the bounds of the array view.

**O.ARRAY_VIEWS_INTEG**

The TOE shall ensure that no application can write to an array view not having array view security attribute ATTR_WRITABLE_VIEW. The TOE shall ensure that an application can only write within the bounds of the array view.

### *5.1.3 SERVICES*

**O.ALARM**

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

**O.CIPHER**

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

**O.RNG**

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

**O.KEY-MNGT**

The TOE shall provide a means to securely manage cryptographic keys (D.APP_KEYs, D.JCS_KEYS and D.CRYPTO). 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 (including the PIN try limit, PIN try counter and states). If the PIN try limit is reached, no further PIN authentication must be allowed. See #.PIN-MNGT for details. This concerns at least the correct authentication of the cardholder and the PIN before having access to protected operations; the observability of the comparison between presented PIN and stored PIN.

*Application Note:*

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

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

### *5.1.4 OBJECT DELETION*

**O.OBJ-DELETION**

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

### *5.1.5 APPLET MANAGEMENT*

**O.DELETION**

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

**O.LOAD**

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

*Application Note:*

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

**O.INSTALL**

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

### 5.1.6 Additional security objectives for the TOE

Four security objectives for the operational environment defined in the PP JCS have been transformed in security objectives for the TOE:

- OE.SCP.IC
- OE.SCP.SUPPORT
- OE.SCP.RECOVERY
- OE.CARD_MANAGEMENT

**O.SCP.SUPPORT**

The TOE shall support the following functionalities:

- It does not allow the TSFs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System.
- It provides secure low-level cryptographic processing to the Java Card System and Global Platform.
- It supports the needs for any update to a single persistent object or class field to be atomic, and a low-level transaction mechanism.
- It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low-level control accesses (segmentation fault detection).

**O.SCP.IC**

The SCP shall possess IC security features. It shall provide all IC security features against physical attacks. 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. 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.RESIDENT_APPLICATION**

The objective ensures the access control to the configuration operations during preperso phase:

In prepersonnalisation the RA shall control the access of Personalizer for product configuration: the loading of sensitive data (locks, patches and ISK keys encrypted with the appropriate key), and keys destructions (MSK and LSK keys) once ISK loaded.

**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.

**O.SECURE_COMPARE**

The TOE shall provide to applet a means to securely compare two byte arrays, i.e. countermeasures against the following attacks: timing attack, comparison loop interrupted and result corrupted.

This objective ensure that no residual information is available from this operation to attackers. The operation of the comparison maintain the confidentiality of the compared arrays.

**O.PATCH_LOADING**

The TOE shall provide a secure patch code loading mechanism. The data to be loaded are encrypted. Once these data loaded, the integrity of the modified code is updated and compared to the provided one in the patch package.

**O.JBox_FW**

The TOE shall provide separation between the non-certified TPL in the JBox and the platform. This separation shall comprise software execution and data access. Security mechanisms have to be implemented to avoid fraudulent usage of the TOE or usage of certain memory regions. The security mechanisms must be designed to always put the TOE in a known and secure state.

**O.FLEXICODE**

The TOE shall provide a secure deletion of modules. The code deleted shall be completly removed: Only the predefined modules can be accessed for deletion, the integrity of code shall be preserved. See #.FLEXICODE for details.

**O.DUMP_PERSO**

The TOE shall provide a secure memory configuration loading and extracting mechanism. When card is in OP_READY state, the data to be extracted are encrypted. The image

dumpped stays encrypted when loaded on another card for a same applet personalization. Once these data loaded, the integrity is compared to the provided one in the image dump.

**O.CLFDB_DECIPHER**

If the SD to be associated with the applet has the Ciphered Load File Data Block privilege, then the card shall support encryption schemes as defined by GlobalPlatform specifications and the SD shall be able to decipher the Ciphered Load File Data Blocks.

*Application Note:*

See **[GP1]** section C.6 This objective refines O.CIPHER.

### *5.1.7 Security Objectives for the TOE with Monotonic counters package*

Package MonotonicCounter defines mechanism for creating a counter that can only be increased.

**O.MTC-CTR-MNGT**

The TOE shall provide a means to securely manage value of the monotonic counter. This concerns the optional package javacardx.security.util of the Java Card platform.

### *5.1.8 Security Objectives for the TOE with PACE package*

The Objectives for PACE Module are refined from ICAO-PACE PP **[PP0068]** in a more generic form in order to be applicable to any application requiring PACE protocol. The eMRTD is kept as application exemple. The naming O.xxx from PP is transformed in O.PACE_xxxx in this ST.

**O.PACE_Data_Integrity**

***Integrity of Data*** The TOE must ensure integrity of the User Data and the TSF-data stored on it by protecting these data against unauthorised modification (physical manipulation and unauthorised modifying). The TOE must ensure integrity of the User Data and the TSF-data during their exchange between the TOE and the terminal connected (and represented by PACE authenticated BIS-PACE) after the PACE Authentication.

**O.PACE_Data_Authenticity**

***Authenticity of Data*** The TOE must ensure authenticity of the User Data and the TSF-data stored on it by enabling verification of their authenticity at the terminal-side. The TOE must ensure authenticity of the User Data and the TSF-data during their exchange between the TOE and the terminal connected (and represented by PACE authenticated BIS-PACE)after the PACE Authentication. It shall happen by enabling such a verification at the terminal-side (at receiving by the terminal) and by an active verification by the TOE itself (at receiving by the TOE).

**O.PACE_Data_Confidentiality**

***Confidentiality of Data*** The TOE must ensure confidentiality of the User Data and the TSF-data by granting read access only to the PACE authenticated BIS-PACE connected. The TOE must ensure confidentiality of the User Data and the TSF-data during their exchange between the TOE and the terminal connected (and represented by PACE authenticated BIS-PACE) after the PACE Authentication.

**O.PACE_Prot_Abuse-Func**

***Protection against Abuse of Functionality*** The TOE must prevent that functions of the TOE, which may not be used in TOE operational phase, can be abused in order (i) to manipulate or to disclose the User Data stored in the TOE, (ii) to manipulate or to disclose the TSF-data stored in the TOE, (iii) to manipulate (bypass, deactivate or modify) soft-coded security functionality of the TOE.

**O.PACE_Prot_Inf_Leak**

***Potection against Information Leakage*** The TOE must provide protection against disclosure of confidential User Data or/and TSF-data stored and/or processed by the electronic document by measurement and analysis of the shape and amplitude of signals or the time between events found by measuring signals on the electromagnetic field, power consumption, clock, or I/O lines, by forcing a malfunction of the TOE and/or by a physical manipulation of the TOE.

**O.PACE_Prot_Phys-Tamper**

***Protection against Physical Tampering*** The TOE must provide protection the confidentiality and integrity of the User Data, the TSF Data, and the MRTD's chip Embedded Software. This includes protection against attacks with high attack potential by means of measuring through galvanic contacts which is direct physical probing on the chips surface except on pads being bonded (using standard tools for measuring voltage and current) or measuring not using galvanic contacts but other types of physical interaction between charges (using tools used in solid-state physics research and IC failure analysis) manipulation of the hardware and its security features, as well as controlled manipulation of memory contents (User Data, TSF Data) with a prior reverse-engineering to understand the design and its properties and functions.

**O.PACE_Prot_Malfunction**

***Protection against Malfunctions*** The TOE must ensure its correct operation. The TOE must prevent its operation outside the normal operating conditions where reliability and secure operation have not been proven or tested. This is to prevent functional errors in the TOE. The environmental conditions may include external energy (especially electromagnetic) fields, voltage (on any contacts), clock frequency or temperature.

**O.PACE_Identification**

***Identification and Authentication of the TOE*** The TOE must provide means to store Initialisation and Pre-Personalisation Data in its non-volatile memory. The Initialisation Data must provide a unique identification of the IC during the manufacturing and the card issuing life cycle phases of the electronic document. The storage of the Pre-Personalisation data includes writing of the Personalisation Agent Key(s).

**O.PACE_AC_Pers**

***Access Control for Personalisation of logical PACE product*** The TOE must ensure that Application data, TOE data and associated TSF data can be written by authorized Personalisation Agents only in personalisation phase. The TOE and Application data requiring PACE usage (e.g. logical electronic document data in EF.DG1 to EF.DG16 for

MRTD) and associated TSF data may be written only during and cannot be changed after personalisation phase.

*Application Note:*

For an ePassport, Application data requiring PACE usage for MRTD is PACE data. For MRTD case, the TOE must ensure that the logical travel document data in EF.DG1 to EF.DG16, the Document Security Object according to LDS **[ICAO-9303]** and the TSF data can be written.

### *5.1.9 Additional security objectives for the TOE with Sensitive Array package*

**O.SENSITIVE_ARRAYS_INTEG**

The TOE shall ensure that only the currently selected applications may have a write access to the integrity-sensitive array object (javacard.framework.SensitiveArrays) created by that application. Any unauthorized modification through physical attacks to that integrity-sensitive array must be detected by the TOE and notified to the application.

### *5.1.10 Additional security objectives for the TOE with Biometrics (MOC) - MODULE*

**O.BIO-MNGT**

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

*Application Note:*

This objective is similar to O.PIN-MNGT. It answers to the same threats.

### *5.1.11 Additional security objectives for the TOE with DBI - MODULE*

**O.DBI-MNGT**

The TOE shall provide a means to securely manage blurred images DBI. This concerns the package idemia.packagedbi of the Java Card platform.

*Application Note:*

This objective is similar to O.PIN-MNGT. It answers to the same threats.

## 5.2 Security Objectives for the operational Environment

This section introduces the security objectives to be achieved by the environment. Four security objectives for the operational environment from the PP JCS have been transformed in security objectives for the TOE:

- OE.SCP.SUPPORT
- OE.SCP.IC
- OE.SCP.RECOVERY
- OE.CARD_MANAGEMENT Not all the security objectives for the operational environment from the protection profile are included in the security target. The OE.CAP_FILE is excluded with justification: Indeed: Instead of the OE.CAP_FILE, this ST adds objectives

O.JBox_FW on the TOE. In that way, applet with the native code is controlled by the platform with O.JBox_FW.

The loading and execution of native code is secured so as to not provide ways to bypass the TSFs of the JCS. These operations are submitted to the access contol and are part of the present evaluation.

These operations are done thru an access controlled by the MPU/MMU, the javacard Firewall, the JBOX Firewall and the OS. In that way, native code embedded in applet is under TOE security control including during post-issuance loading.

Some security objectives for the operational environment has been transformed in security objectives for the TOE, the rationale is presented in section section 2.2.. Other security objectives for the operational environment have been added (see section 5.2.1).

### *5.2.1 Security Objectives for the operational Environment for PACE package*

**OE.PACE_Prot_Logical_Data**

***Protection of TOE and applicative data*** The inspection system of the applicative entity (e.g. receiving State or Organisation) ensures the confidentiality and integrity of the data read from the TOE and applicative data (e.g. logical electronic document). The inspection system will prevent eavesdropping to their communication with the TOE before secure messaging is successfully established.

**OE.PACE_Personalisation**

***Personalisation of TOE and application data for PACE*** The Issuer must ensure that the Personalisation Agents acting on his behalf (i) establish the correct identity of the applicative user (e.g. electronic document holder) and create the accurate applicative data* and write them in TOE. Note: in the specific case of MRTD, accurate applicative data are biographical data for the electronic document), (ii) biometric reference data of the electronic document holder, the initial TSF data, (the Document Security Object defined in **[PKI]** (in the role of a DS).

**OE.PACE_Terminal**

***Terminal operating*** The terminal operators must operate their terminals as follows: 1.) The related terminals (basic inspection systems, cf. above) are used by terminal operators and by electronic document holders as defined in **[ICAO-9303]**. 2.) The related terminals implement the terminal parts of the PACE protocol **[ICAO-9303]** part 11, of the Passive Authentication **[ICAO-9303]** part 11 (by verification of the signature of the Document Security Object) and use them in this order. The PACE terminal uses randomly and (almost) uniformly selected nonces, if required by the protocols (for generating ephemeral keys for Diffie-Hellmann). 3.) The related terminals need not to use any own credentials. 4.) The related terminals and their environment must ensure confidentiality and integrity of respective data handled by them (e.g. confidentiality of the PACE passwords, integrity of PKI certificates, etc.), where it is necessary for a secure operation of the TOE according to the current ST.

**OE.PACE_User_Obligations**

***User Obligations*** The application user (e.g. electronic document holder) may reveal, if necessary, his or her verification values of the PACE password to an authorized person or device who definitely act according to respective regulations and are trustworthy.

### *5.2.2 Miscellaneous*

**OE.VERIFICATION**

All the bytecodes shall be verified at least once, before the loading, before the installation or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. See #.VERIFICATION for details. Additionally, the applet shall follow all the recommendations, if any, mandated in the platform guidance for maintaining the isolation property of the platform.

*Application Note:*

Constraints to maintain the isolation property of the platform are provided by the platform developer in application development guidance. The constraints apply to all application code loaded in the platform. The OE concerns the applets loaded without JBox API, these applets do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM22], §3.3) outside the API. The platform allows the loading of IDEMIA proprietary bytecodes in evaluated applets. Then the evaluation lab does the verifications.

**OE.CODE-EVIDENCE**

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

*Application Note:*

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

**OE.CAP_FILE**

No CAP file loaded post-issuance (without using JBox mechanism) shall contain native methods.

**OE.CLFDB_ENC**

When the associated Security Domain has the Ciphered Load File Data Block privilege, the Load File Data Block shall be encrypted securely by a trusted SD provider.

*Application Note:*

See **[GP1]** section C.6

## 5.3 Security Objectives Rationale

### *5.3.1 Threats*

#### 5.3.1.1 CONFIDENTIALITY

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

As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate 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,, O.RNG). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys, PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT, O.TRANSACTION, O.DBI-MNGT). If the PIN/BIO class of the Java Card API is used, the objective (O.FIREWALL) shall contribute in covering this threat by controlling the sharing of the global PIN between the applets.

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

An applet might share data buffer with another applet using array views without the array view security attribute ATTR_READABLE_VIEW. The disclosure of data of the applet creating the array view is prevented by the security object O.ARRAY_VIEWS_CONFID.

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

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

applications are also harmless because of the objective O.NATIVE, but the O.JBox_FW ensures that the native code is not harmless to javacard platform. The (#.VERIFICATION) security aspect is addressed in this ST by the objective for the environment OE.VERIFICATION.

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

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

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

#### 5.3.1.2 INTEGRITY

**T.INTEG-APPLI-CODE** This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective O.NATIVE and the added objective O.JBox_FW ensures that the native code is not harmless to javacard platform. 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, O.RNG). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys and PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (O.KEY-MNGT, O.PIN-MNGT, O.BIO-MNGT, O.TRANSACTION, O.DBI-MNGT). If the PIN/BIO class of the Java Card API is used, the objective (O.FIREWALL) is also concerned. Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The integrity of the information stored in that buffer is ensured by the objective O.GLOBAL_ARRAYS_INTEG. An applet might share data buffer with another applet using array views without the array view security attribute ATTR_WRITABLE_VIEW. The integrity of data of the applet creating the array view is ensured by the security objective O.ARRAY_VIEWS_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. This threat is countered by the security objective for monotonic counter management (O.MTC-CTR-MNGT) such that value of the monotonic counter will be protected against any unauthorized change.

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

**T.INTEG-JCS-CODE** This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the

intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective O.NATIVE. But the O.JBox_FW ensures that the native code is not harmless to javacard platform. 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 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.

#### 5.3.1.3 IDENTITY USURPATION

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

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

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

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

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

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

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

#### 5.3.1.4 UNAUTHORIZED EXECUTION

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

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

**T.NATIVE** T.NATIVE This threat is countered by O.NATIVE which ensures that a Java Card applet can only access native methods indirectly that is, through an API. This threat is also countered by O.JBox_FW which ensures that the native code is not harmless. OE.CAP_FILE also covers this threat by ensuring that CAP files containing native code applets shall be loaded in post-issuance without using JBox mechanism. In addition to this, the bytecode verifier also prevents the program counter of an applet to jump into a piece of native code by confining the control flow to the currently executed method (OE.VERIFICATION).

#### 5.3.1.5 DENIAL OF SERVICE

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

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

It should be noticed that, for what relates to CPU usage, the Java Card platform is single-threaded and it is possible for an ill-formed application to monopolize the CPU. However, a smart card can be physically interrupted (card removal or hardware reset) and most CADs implement a timeout policy that prevent them from being blocked should a card fails to answer. That point is out of scope of this Security Target, though.

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

#### 5.3.1.6 CARD MANAGEMENT

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

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

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

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

#### 5.3.1.7 SERVICES

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

#### 5.3.1.8 MISCELLANEOUS

**T.PHYSICAL** Covered by O.SCP.IC. Physical protections rely on the underlying platform and are therefore an environmental issue. O.SENSITIVE_ARRAYS_INTEG requires the TOE to detect and notify the application if any unauthorized modification of the integrity-sensitive array object through physical attacks occurred.

#### 5.3.1.9 Additional threats

**T.CONFIGURATION** This threat is covered by O.RESIDENT_APPLICATION.

This objective ensures that any operation in the prepersonalisation phase need authentication, it ensures also that D.CONFIG or D.PERSO_DUMP is loaded protected from theft and modification such as an attacker cannot observe or modify configuration information exchanged between the TOE and its environment.

**T.CONF_DATA_APPLET** This threat is covered by the O.SECURE_COMPARE security objective.

If an attacker tries to catch confidential information "D.ARRAY", the objective O.SECURE_COMPARE ensures that no residual information is available to the attacker.

**T.PATCH_LOADING** This threat is covered by O.PATCH_LOADING security objective.

If an attacker tries to avoid the loading of a patch or alter a patch (during loading or once loaded), O.PATCH_LOADING ensures trustable identification and authentication (static signature) data of the loaded patch are returned by the TOE. This information enables to check the presence of the genuine patch. Moreover, O.PATCH_LOADING, ensures authentication of the entity loading the patch before the patch is loaded in the TOE. This objective ensures patch loading can only be performed with a limited access.

**T.JBox_BORDER** This threat is covered by the O.JBox_FW security objective. It ensures that the TPL in the JBox and the data belonging to this TPL is completed sealed off from the rest of the platform. Due to this separation, the TPL in the JBox cannot harm the code and data outside the JBox. The execution of native code is supported by O.NATIVE.

**T.PERSO_DUMP** This threat is covered by O.RESIDENT_APPLICATION and O.DUMP_PERSO security objectives.

If an attacker tries to avoid the loading perso dump or alter the image (during loading or once loaded), O.DUMP_PERSO ensures confidential and unmodified data of the extracted and loaded image. This information enables to check the presence of the genuine image. The memory blocks read are encrypted with DSK key. The TOE can dechipher the loaded data in personalization with DSK key. Moreover, O.RESIDENT_APPLICATION, ensures the access control to the configuration operations during pre-personalization phase. The card must be in pre-personalization life phase OP_READY state to extract data.

**T.CLFDB-DISC** This threat is covered by O.CLFDB_DECIPHER security objective. The CLFDB are sent enciphered in the way to be protected in confidentiality (OE.CLFDB_ENC)and the TOE need to decipher the data by O.CLFDB_DECIPHER.

#### 5.3.1.10 Additional threats for FLEXICODE package

**T.FLEXICODE** This threat is covered by the O.FLEXICODE that ensures the secure deletion of the code and to maintain TOE integrity. The access to the FLEXICODE functionnality is controlled by O.RESIDENT_APPLICATION.

#### 5.3.1.11 Threats for PACE package

**T.PACE_Skimming** The threat T.PACE_Skimming addresses accessing the User Data (stored on the TOE or transferred between the TOE and the terminal) using the TOE's contactless/contact interface. This threat is countered by the security objectives O.PACE_Data_Integrity, O.PACE_Data_Authenticity and O.PACE_Data_Confidentiality through the PACE authentication. The objective OE.PACE_User_Obligations ensures that a PACE session can only be established either by the application user itself (e.g. electronic document holder for MRTD) or by an authorised person or device, and, hence, cannot be captured by an attacker.

**T.PACE_Eavesdropping** The threat T.PACE_Eavesdropping addresses listening to the communication between the TOE and a rightful terminal in order to gain the User Data transferred there. This threat is countered by the security objective O.PACE_Data_Confidentiality through a trusted channel based on the PACE authentication.

**T.PACE_Forgery** The threat T.PACE_Forgery addresses the fraudulent, complete or partial alteration of the User Data or/and TSF-data stored on the TOE or/and exchanged between the TOE and the terminal. The security objective O.PACE_AC_Pers requires the TOE to limit the write access for the TOE and applicative data to the trustworthy Personalisation Agent (cf. OE.PACE_Personalisation). The TOE will protect the integrity and authenticity of the stored and exchanged User Data or/and TSF-data as aimed by the security objectives

O.PACE_Data_Integrity and O.PACE_Data_Authenticity, respectively. The objectives O.PACE_Prot_Phys-Tamper and O.PACE_Prot_Abuse-Func contribute to protecting integrity of the User Data or/and TSF-data stored on the TOE. A terminal operator operating his terminals according to OE.PACE_Terminal to contribute to secure exchange between the TOE and the terminal.

**T.PACE_Abuse-Func** The threat T.PACE_Abuse-Func addresses attacks of misusing TOE's functionality to manipulate or to disclosure the stored User- or TSF-data as well as to disable or to bypass the soft-coded security functionality. The security objective O.PACE_Prot_Abuse-Func ensures that the usage of functions having not to be used in the operational phase is effectively prevented.

**T.PACE_Information_Leakage** The threat T.PACE_Information_Leakage is typical for integrated circuits like smart cards under direct attack with high attack potential. The protection of the TOE against this threat is obviously addressed by the directly related security objective O.PACE_Prot_Inf_Leak

**T.PACE_Phys-Tamper** The threat T.PACE_Phys-Tamper is typical for integrated circuits like smart cards under direct attack with high attack potential. The protection of the TOE against this threat is obviously addressed by the directly related security objective O.PACE_Prot_Phys-Tamper

**T.PACE_Malfunction** The threat T.PACE_Malfunction is typical for integrated circuits like smart cards under direct attack with high attack potential. The protection of the TOE against this threat is obviously addressed by the directly related security objective O.PACE_Prot_Malfunction

### *5.3.2 Organisational Security Policies*

#### 5.3.2.1 OSP for Verification

**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, and by the security

objective for the TOE O.LOAD which shall ensure that the loading of a package into the card is safe.

#### 5.3.2.2 Organisational Security policies for PACE package

**P.PACE_Manufact** The OSP P.PACE Manufact requires a unique identification of the IC by means of the Initialization Data and the writing of the Pre-personalisation Data as being fulfilled by O.PACE_Identification.

**P.PACE_Pre-operational** The OSP P.PACE_Pre-operational "Pre-operational handling of the electronic document" is enforced by the following security objectives: O.PACE_Identification "Identification of the TOE" is affine to the OSP's property 'traceability before the operational phase'; O.PACE_AC_Pers and OE.PACE_Personalisation together enforce the OSP's properties 'correctness of the User- and the TSF-data stored' and 'authorisation of Personalisation Agents'.

**P.PACE_Terminal** The OSP P.PACE_Terminal "Abilities and trustworthiness of terminals" is countered by the security objective OE.PACE_Terminal enforces the terminals to perform the terminal part of the PACE protocol.

**P.PACE_Personalisation** The OSP P.Personalisation addresses the (i) the enrolment of the logical electronic document by the Personalisation Agent as described in the security objective for the TOE environment OE.PACE_Personalisation, and (ii) the access control for the user data and TSF data as described by the security objective O.PACE_AC_Pers. Note the manufacturer equips the TOE with the Personalisation Agent Key(s) according to O.PACE_Identification "Identification of the TOE". The security objective O.PACE_AC_Pers limits the management of TSF data and the management of TSF to the Personalisation Agent.

#### 5.3.2.3 OSP for CLFDB

**OSP.CLFDB_ENC** This policy is upheld by the security objective of the environment OE.CLFDB_ENC which guarantees that the block sent are securely enciphered by a trusted SD provider.

### *5.3.3 Assumptions*

#### 5.3.3.1 Asumptions for Verification

**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.

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

#### 5.3.3.2 Asumptions for PACE package

**A.PACE_Insp_Sys** A.PACE_Insp_Sys is covered by OE.PACE_Prot_Logical_Data requiring the Inspection System to protect the TOE and application data (e.g. the logical electronic document data) during the transmission and the internal handling.

### *5.3.4 SPD and Security Objectives*

| Threats | Security Objectives | Rationale |
|---|---|---|
| T.CONFID-APPLI-DATA | OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.REALLOCATION, O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD_MANAGEMENT, O.RNG, O.BIO-MNGT, O.ARRAY_VIEWS_CONFID, O.DBI-MNGT | Section 5.3.1 |
| T.CONFID-JCS-CODE | OE.VERIFICATION, O.CARD_MANAGEMENT, O.JBox_FW, O.NATIVE | Section 5.3.1 |
| T.CONFID-JCS-DATA | OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM, O.SCP.RECOVERY, O.SCP.SUPPORT, O.CARD_MANAGEMENT | Section 5.3.1 |
| T.INTEG-APPLI-CODE | OE.VERIFICATION, OE.CODE-EVIDENCE, O.CARD_MANAGEMENT, O.JBox_FW, O.NATIVE | Section 5.3.1 |
| T.INTEG-APPLI-CODE.LOAD | O.LOAD, OE.CODE-EVIDENCE, O.CARD_MANAGEMENT | Section 5.3.1 |
| T.INTEG-APPLI-DATA | OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.GLOBAL_ARRAYS_INTEG, O.ALARM, O.TRANSACTION, O.CIPHER, O.PIN-MNGT, O.KEY-MNGT, O.REALLOCATION, O.SCP.RECOVERY, O.SCP.SUPPORT, OE.CODE-EVIDENCE, O.CARD_MANAGEMENT, O.RNG, O.BIO-MNGT, O.ARRAY_VIEWS_INTEG, O.MTC-CTR-MNGT, O.DBI-MNGT | Section 5.3.1 |
| T.INTEG-APPLI-DATA.LOAD | O.LOAD, OE.CODE-EVIDENCE, O.CARD_MANAGEMENT | Section 5.3.1 |

| | | |
|---|---|---|
| T.INTEG-JCS-CODE | OE.VERIFICATION, OE.CODE-EVIDENCE, O.CARD_MANAGEMENT, O.JBox_FW, O.NATIVE | Section 5.3.1 |
| T.INTEG-JCS-DATA | OE.VERIFICATION, O.SID, O.OPERATE, O.FIREWALL, O.ALARM, O.SCP.RECOVERY, O.SCP.SUPPORT, OE.CODE-EVIDENCE, O.CARD_MANAGEMENT | Section 5.3.1 |
| T.SID.1 | O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.INSTALL, O.SID, O.CARD_MANAGEMENT | Section 5.3.1 |
| T.SID.2 | O.SID, O.OPERATE, O.FIREWALL, O.INSTALL, O.SCP.RECOVERY, O.SCP.SUPPORT | Section 5.3.1 |
| T.EXE-CODE.1 | OE.VERIFICATION, O.FIREWALL | Section 5.3.1 |
| T.EXE-CODE.2 | OE.VERIFICATION | Section 5.3.1 |
| T.NATIVE | OE.VERIFICATION, O.NATIVE, O.JBox_FW, OE.CAP_FILE | Section 5.3.1 |
| T.RESOURCES | O.INSTALL, O.OPERATE, O.RESOURCES, O.SCP.RECOVERY, O.SCP.SUPPORT | Section 5.3.1 |
| T.DELETION | O.DELETION, O.CARD_MANAGEMENT | Section 5.3.1 |
| T.INSTALL | O.INSTALL, O.LOAD, O.CARD_MANAGEMENT | Section 5.3.1 |
| T.OBJ-DELETION | O.OBJ-DELETION | Section 5.3.1 |
| T.PHYSICAL | O.SCP.IC, O.SENSITIVE_ARRAYS_INTEG | Section 5.3.1 |
| T.CONFIGURATION | O.RESIDENT_APPLICATION | Section 5.3.1 |
| T.CONF_DATA_APPLET | O.SECURE_COMPARE | Section 5.3.1 |
| T.PATCH_LOADING | O.PATCH_LOADING | Section 5.3.1 |
| T.JBox_BORDER | O.JBox_FW, O.NATIVE | Section 5.3.1 |
| T.PERSO_DUMP | O.RESIDENT_APPLICATION, O.DUMP_PERSO | Section 5.3.1 |
| T.CLFDB-DISC | O.CLFDB_DECIPHER, OE.CLFDB_ENC | Section 5.3.1 |
| T.FLEXICODE | O.RESIDENT_APPLICATION, O.FLEXICODE | Section 5.3.1 |
| T.PACE_Skimming | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality, OE.PACE_User_Obligations | Section 5.3.1 |
| T.PACE_Eavesdropping | O.PACE_Data_Confidentiality | Section 5.3.1 |
| T.PACE_Forgery | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, | Section 5.3.1 |

| | | |
|---|---|---|
| | O.PACE_Prot_Abuse-Func, O.PACE_Prot_Phys-Tamper, OE.PACE_Terminal, OE.PACE_Personalisation, O.PACE_AC_Pers | |
| T.PACE_Abuse-Func | O.PACE_Prot_Abuse-Func | Section 5.3.1 |
| T.PACE_Information_Leakage | O.PACE_Prot_Inf_Leak | Section 5.3.1 |
| T.PACE_Phys-Tamper | O.PACE_Prot_Phys-Tamper | Section 5.3.1 |
| T.PACE_Malfunction | O.PACE_Prot_Malfunction | Section 5.3.1 |

**Table 5 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, T.JBox_BORDER | |
| 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.ARRAY_VIEWS_CONFID | T.CONFID-APPLI-DATA | |
| O.ARRAY_VIEWS_INTEG | T.INTEG-APPLI-DATA | |
| 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.RNG | 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.TRANSACTION | T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA | |
| O.OBJ-DELETION | T.OBJ-DELETION | |

| | | |
|---|---|---|
| O.DELETION | T.DELETION | |
| O.LOAD | T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA.LOAD, T.INSTALL | |
| O.INSTALL | T.SID.1, T.SID.2, T.RESOURCES, T.INSTALL | |
| O.SCP.SUPPORT | T.CONFID-APPLI-DATA, T.CONFID-JCS-DATA, T.INTEG-APPLI-DATA, T.INTEG-JCS-DATA, T.SID.2, T.RESOURCES | |
| 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.RESIDENT_APPLICATION | T.CONFIGURATION, T.PERSO_DUMP, T.FLEXICODE | |
| O.CARD_MANAGEMENT | T.CONFID-APPLI-DATA, T.CONFID-JCS-CODE, T.CONFID-JCS-DATA, T.INTEG-APPLI-CODE, T.INTEG-APPLI-CODE.LOAD, T.INTEG-APPLI-DATA, T.INTEG-APPLI-DATA.LOAD, T.INTEG-JCS-CODE, T.INTEG-JCS-DATA, T.SID.1, T.DELETION, T.INSTALL | |
| O.SECURE_COMPARE | T.CONF_DATA_APPLET | |
| O.PATCH_LOADING | T.PATCH_LOADING | |
| O.JBox_FW | T.CONFID-JCS-CODE, T.INTEG-APPLI-CODE, T.INTEG-JCS-CODE, T.NATIVE, T.JBox_BORDER | |
| O.FLEXICODE | T.FLEXICODE | |
| O.DUMP_PERSO | T.PERSO_DUMP | |
| O.CLFDB_DECIPHER | T.CLFDB-DISC | |
| O.MTC-CTR-MNGT | T.INTEG-APPLI-DATA | |
| O.PACE_Data_Integrity | T.PACE_Skimming, T.PACE_Forgery | |
| O.PACE_Data_Authenticity | T.PACE_Skimming, T.PACE_Forgery | |
| O.PACE_Data_Confidentiality | T.PACE_Skimming, T.PACE_Eavesdropping | |
| O.PACE_Prot_Abuse-Func | T.PACE_Forgery, T.PACE_Abuse-Func | |
| O.PACE_Prot_Inf_Leak | T.PACE_Information_Leakage | |
| O.PACE_Prot_Phys-Tamper | T.PACE_Forgery, T.PACE_Phys-Tamper | |
| O.PACE_Prot_Malfunction | T.PACE_Malfunction | |
| O.PACE_Identification | | |
| O.PACE_AC_Pers | T.PACE_Forgery | |

| | | |
|---|---|---|
| O.SENSITIVE_ARRAYS_INTEG | T.PHYSICAL | |
| O.BIO-MNGT | T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA | |
| O.DBI-MNGT | T.CONFID-APPLI-DATA, T.INTEG-APPLI-DATA | |
| OE.PACE_Prot_Logical_Data | | |
| OE.PACE_Personalisation | T.PACE_Forgery | |
| OE.PACE_Terminal | T.PACE_Forgery | |
| OE.PACE_User_Obligations | T.PACE_Skimming | |
| 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, T.INTEG-JCS-DATA | |
| OE.CAP_FILE | T.NATIVE | |
| OE.CLFDB_ENC | T.CLFDB-DISC | |

**Table 6 Security Objectives and Threats - Coverage**

| Organisational Security Policies | Security Objectives | Rationale |
|---|---|---|
| OSP.VERIFICATION | OE.VERIFICATION, OE.CODE-EVIDENCE, O.LOAD | Section 5.3.2 |
| P.PACE_Manufact | O.PACE_Identification | Section 5.3.2 |
| P.PACE_Pre-operational | O.PACE_Identification, O.PACE_AC_Pers, OE.PACE_Personalisation | Section 5.3.2 |
| P.PACE_Terminal | OE.PACE_Terminal | Section 5.3.2 |
| P.PACE_Personalisation | OE.PACE_Personalisation, O.PACE_AC_Pers, O.PACE_Identification | Section 5.3.2 |
| OSP.CLFDB_ENC | OE.CLFDB_ENC | Section 5.3.2 |

**Table 7 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.ARRAY_VIEWS_CONFID | | |
| O.ARRAY_VIEWS_INTEG | | |
| O.ALARM | | |
| O.CIPHER | | |
| O.RNG | | |
| O.KEY-MNGT | | |
| O.PIN-MNGT | | |
| O.TRANSACTION | | |
| O.OBJ-DELETION | | |
| O.DELETION | | |
| O.LOAD | OSP.VERIFICATION | |
| O.INSTALL | | |
| O.SCP.SUPPORT | | |
| O.SCP.IC | | |
| O.SCP.RECOVERY | | |
| O.RESIDENT_APPLICATION | | |
| O.CARD_MANAGEMENT | | |
| O.SECURE_COMPARE | | |
| O.PATCH_LOADING | | |
| O.JBox_FW | | |
| O.FLEXICODE | | |
| O.DUMP_PERSO | | |
| O.CLFDB_DECIPHER | | |
| O.MTC-CTR-MNGT | | |
| O.PACE_Data_Integrity | | |
| O.PACE_Data_Authenticity | | |
| O.PACE_Data_Confidentiality | | |
| O.PACE_Prot_Abuse-Func | | |

| | | |
|---|---|---|
| O.PACE_Prot_Inf_Leak | | |
| O.PACE_Prot_Phys-Tamper | | |
| O.PACE_Prot_Malfunction | | |
| O.PACE_Identification | P.PACE_Manufact, P.PACE_Pre-operational, P.PACE_Personalisation | |
| O.PACE_AC_Pers | P.PACE_Pre-operational, P.PACE_Personalisation | |
| O.SENSITIVE_ARRAYS_INTEG | | |
| O.BIO-MNGT | | |
| O.DBI-MNGT | | |
| OE.PACE_Prot_Logical_Data | | |
| OE.PACE_Personalisation | P.PACE_Pre-operational, P.PACE_Personalisation | |
| OE.PACE_Terminal | P.PACE_Terminal | |
| OE.PACE_User_Obligations | | |
| OE.VERIFICATION | OSP.VERIFICATION | |
| OE.CODE-EVIDENCE | OSP.VERIFICATION | |
| OE.CAP_FILE | | |
| OE.CLFDB_ENC | OSP.CLFDB_ENC | |

**Table 8 Security Objectives and OSPs - Coverage**

| Assumptions | Security Objectives for the Operational Environment | Rationale |
|---|---|---|
| A.VERIFICATION | OE.VERIFICATION, OE.CODE-EVIDENCE | Section 5.3.3 |
| A.CAP_FILE | OE.CAP_FILE | Section 5.3.3 |
| A.PACE_Insp_Sys | OE.PACE_Prot_Logical_Data | Section 5.3.3 |

**Table 9 Assumptions and Security Objectives for the Operational Environment - Coverage**

| Security Objectives for the Operational Environment | Assumptions | Rationale |
|---|---|---|
| OE.PACE_Prot_Logical_Data | A.PACE_Insp_Sys | |
| OE.PACE_Personalisation | | |
| OE.PACE_Terminal | | |
| OE.PACE_User_Obligations | | |
| OE.VERIFICATION | A.VERIFICATION | |
| OE.CODE-EVIDENCE | A.VERIFICATION | |

| | | |
|---|---|---|
| OE.CAP_FILE | A.CAP_FILE | |
| OE.CLFDB_ENC | | |

**Table 10 Security Objectives for the Operational Environment and Assumptions - Coverage**

# 6 Extended Requirements

## 6.1 Extended Families

### *6.1.1 Extended Family FCS_RND - Generation of random numbers*

#### 6.1.1.1 Description

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

#### 6.1.1.2 Extended Components

**Extended Component FCS_RND.1**

*Description*

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

*Definition*

| **FCS_RND.1 Quality metric for random numbers** |
|---|

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

Dependencies: No dependencies.

### *6.1.2 Extended Family FMT_LIM - Limited capabilities*

#### 6.1.2.1 Description

The family FMT_LIM describes the functional requirements for the test features of the TOE. The new functional requirements were defined in the class FMT because this class addresses the management of functions of the TSF. The examples of the technical mechanism used in the TOE show that no other class is appropriate to address the specific issues of preventing abuse of functions by limiting the capabilities of the functions and by limiting their availability.

#### 6.1.2.2 Extended Components

**Extended Component FMT_LIM.1**

*Description*

*Definition*

| **FMT_LIM.1 Limited capabilities** |
|---|

**FMT_LIM.1.1** The TSF shall be designed in a manner that limits their capabilities so that in conjunction with 'Limited availability (FMT_LIM.2)' the following policy is enforced [assignment: Limited capability and availability policy]

Dependencies: No dependencies.

**Extended Component FMT_LIM.2**

*Description*

*Definition*

| **FMT_LIM.2 Limited capabilities** |
|---|

**FMT_LIM.2.1** The TSF shall be designed in a manner that limits their availability so that in conjunction with 'Limited capabilities (FMT_LIM.1)' the following policy is enforced [assignment: Limited capability and availability policy]

Dependencies: No dependencies.

### *6.1.3 Extended Family FPT_EMS - TOE Emanation*

#### 6.1.3.1 Description

The additional family FPT_EMS (TOE Emanation) of the Class FPT (Protection of the TSF) is defined here to describe the IT security functional requirements of the TOE. The TOE shall prevent attacks against the SCD and other secret data where the attack is based on external observable physical phenomena of the TOE. Examples of such attacks are evaluation of TOE?s electromagnetic radiation, simple power analysis (SPA), differential power analysis (DPA), timing attacks, radio emanation etc. This family describes the functional requirements for the limitation of intelligible emanations. The family FPT_EMS belongs to the Class FPT because it is the class for TSF protection. Other families within the Class FPT do not cover the TOE emanation.

#### 6.1.3.2 Extended Components

**Extended Component FPT_EMS.1**

*Description*

This family defines requirements to mitigate intelligible emanations.

FPT_EMS.1 TOE Emanation has two constituents:

- FPT_EMS.1.1 Limit of Emissions requires to not emit intelligible emissions enabling access to TSF data or user data.
- FPT_EMS.1.2 Interface Emanation requires to not emit interface emanation enabling access to TSF data or user data.

*Definition*

| **FPT_EMS.1 TOE Emanation** |
|---|

**FPT_EMS.1.1** The TOE shall not emit [assignment: types of emissions] in excess of [assignment: specified limits] enabling access to [assignment: list of types of TSF data].

**FPT_EMS.1.2** The TSF shall ensure [assignment: type of users] are unable to use the following interface [assignment: type of connection] to gain access to [assignment: list of types of TSF data].

Dependencies: No dependencies.

### *6.1.4 Extended Family FCS_RNG - Random Number Generation*

#### 6.1.4.1 Description

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

#### 6.1.4.2 Extended Components

**Extended Component FCS_RNG.1**

*Description*

A physical random number generator (RNG) produces the random number by a noise source based on physical random processes. A non-physical true RNG uses a noise source based on non-physical random processes like human interaction (key strokes, mouse movement). A deterministic RNG uses a random seed to produce a pseudorandom output. A hybrid RNG combines the principles of physical and deterministic RNGs.

## FCS_RNG.1 Random Number Generation

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

**FCS_RNG.1.2** The TSF shall provide random numbers that meet [assignment: a defined quality metric].

Dependencies: No dependencies.

# 7 Security Requirements

## 7.1 Security Functional Requirements

This section states the security functional requirements for the Java Card System - Open configuration. For readability and for compatibility with the original Java Card System Protection Profile Collection - Standard 2.2 Configuration **[PP0035]**, requirements are arranged into groups. All the groups defined in the table below apply to this Security Target.

| Group | Description |
|---|---|
| Core with Logical Channels (CoreG_LC) | The CoreG_LC contains the requirements concerning the runtime environment of the Java Card System implementing logical channels. This includes the firewall policy and the requirements related to the Java Card API. Logical channels are a Java Card specification version 2.2 feature. This group is the union of requirements from the Core (CoreG) and the Logical channels (LCG) groups defined in **[PP0035]** (cf. Java Card System Protection Profile Collection **[PP0035]**). |
| Installation (InstG) | The InstG contains the security requirements concerning the installation of post-issuance applications. It does not address card management issues in the broad sense, but only those security aspects of the installation procedure that are related to applet execution. |
| Applet deletion (ADELG) | The ADELG contains the security requirements for erasing installed applets from the card, a feature introduced in Java Card specification version 2.2. |
| Object deletion (ODELG) | The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism. This is a Java Card specification version 2.2 feature. |
| Secure carrier (CarG) | The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing 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, 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.

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

| Object | Description |
|---|---|
| O.APPLET | Any installed applet, its code and data |

| | |
|---|---|
| O.CODE_PKG | The code of a package, including all linking information. On the Java Card platform, a package is the installation unit |
| O.JAVAOBJECT | Java class instance or array. It should be noticed that KEYS, PIN, arrays and applet instances are specific objects in the Java programming language |
| O.TPL_Content | The code and data elements of the TPL residing in the JBox. |
| O.Platform_Content | Any code and data elements not assigned to the TPL. |
| O.JBox_Interface | All platform interfaces ie services offered by the platform and available to S.TPL: Fetch method parameter (POP), Push return value (PUSH), Read array, Write array, Exit from TPL execution |
| O.Platform_Interface | All interfaces available on the platform excepts those defined in 0.JBox_Interface. |

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 |
|---|---|
| Active Applets | The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels. |
| Applet Selection Status | "Selected" or "Deselected". |
| Applet's version number | The version number of an applet (package) indicated in the export file. |
| Context | Package AID or "Java Card RE". |
| COD Context attribute | Delimits the space occupied in volatile memory by the data of the CLEAR_ON_DESELECT transient arrays of a package |
| COR Context attribute | Delimits the space occupied in volatile memory by the data of the CLEAR_ON_RESET transient arrays of a package |
| Current Frame Context | The lower and upper Boundary of the local variables area on the stack frame for a method and the lower and upper Boundary of the operand stack area on the stack frame for a method |
| Currently Active Context | Package AID or "Java Card 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 (*) or CLEAR_ON_RESET |
| Object Boundary | Delimits the space occupied by an object in the heap |
| 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. |
| Package Boundary | Delimits the space occupied by the code and the static fields of a package |
| Program Counter | Position of the next Bytecode to executed |
| 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, Array View, Java Card RE entry point or global array. |
| Stack Pointer | Position of the next free slot in the stack |
| Static Fields | Static fields of a package |
| 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_DESELECT 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.ARRAY_LENGTH (O.JAVAOBJECT, field) | Get length of an array component |
| OP.ARRAY_AASTORE(O.JAVAOBJECT, field) | Store into reference array component |
| OP.CREATE (Sharing, LifeTime) (*) | Creation of an object (new, makeTransient or createArrayView 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.FLOW (O.CODE_PKG) | Any operation that modify the execution flow |
| OP.IMPORT_KEY | Import of the keys |
| OP.INSTANCE_FIELD (O.JAVAOBJECT, field) | Read/Write a field of an instance of a class in the Java programming language. |
| OP.INVK_INTERFACE (O.JAVAOBJECT, method, arg1,...) | Invoke an interface method. |
| OP.INVK_VIRTUAL (O.JAVAOBJECT, method, arg1,...) | Invoke a virtual method (either on a class instance or an array object). |
| 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.LOCAL_STACK_ACCESS (...) | Any operation that read or write the local stack |
| OP.OPERAND_STACK_ACCESS (...) | Any operation that push or pop items on the operand stack |
| OP.PUT (S1,S2,I) | Transfer a piece of information I from S1 to S2. |
| OP.STATIC_FIELD (O.CODE_PKG, field) | Read/Write a static field of a class in the JAVA programming language |
| 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). |
| Cardholder Authentication | Authentication of the cardholder |
| U.Card_Issuer authentication | Authentication of U.Card_Issuer |

(*) For this operation, there is no accessed object. This rule enforces that shareable transient objects are not allowed, except some objects, such as COR. For more information refer to the

Java Doc **[JCAPI]**. For instance, during the creation of an object, the JavaCardClass attribute's value is chosen by the creator.

### *7.1.1 CoreG_LC Security Functional Requirements*

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

#### 7.1.1.1 Application Programming Interface

The following SFRs are related to the Java Card API.

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

**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:**

| **Cryptographic key generation algorithm** | **Cryptographic key size** | **List of standards** |
|---|---|---|
| **TDES** | **112 bits or 168 bits** | **FIPS PUB 46-3 (ANSI X3.92), FIPS PUB 81** |
| **ECKeyP** | **from 192 to 521 bits** | **IEEE Std 1363a-2004 [IEEE]** |
| **XECKeyP** | **from 192 to 521 bits** | **BRAINPOOL (RFC 5639) [BRAINPL]** |
| **AES** | **from 128 to 256 bits with a step of 64 bits** | **FIPS PUB 197** |
| **GP Keys - TDES (ECB)** | **112 bits** | **GP 2.3** |
| **GP Keys – AES (CBC)** | **128, 192, 256 bits** | **GP 2.3** |
| **GP Keys – AES (ECB)** | **128, 256 bits** | **GP 2.3** |

.

*Application Note:*

- The keys can be generated and diversified in accordance with **[JCAPI]** specification in classes KeyPair (at least Session key generation) and RandomData.
- For Elliptic curves ECDSA and ECDH, the TOE shall provide a subset of cryptographic operations ECC Brainpool with key XECKeyP: the Named curves and their domain parameters are introduced in package javacard.security.NamedParameterSpec:

- Named curves Permanently supported by the TOE as standard curves are:
  - BRAINPOOLP256R1
  - BRAINPOOLP384R1
  - SECP256R1
  - SECP384R1
- Optionally supported (i.e. to be loaded in personalization)
  - BRAINPOOLP192R1, BRAINPOOLP192T1, BRAINPOOLP224R1, BRAINPOOLP224T1, BRAINPOOLP256T1, BRAINPOOLP320R1, BRAINPOOLP320T1, BRAINPOOLP384T1, BRAINPOOLP512R1, BRAINPOOLP512T1
  - FRP256V1
  - SECP192R1, SECP224R1, SECP521R1

## 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 **the keys are reset with the method clearKey()** that meets the following: **"Java Card API" specification [JCAPI]. The keys used in class ISOSecureMessaging (Package "com.oberthurcs.javacard.utilSM") [AGD_OPE] are classes Key that meets the following: "Java Card API" specification [JCAPI]. The methods 'reset' and 'setKeyFormat' call the method key.clearKey() for clearing the value of each key**.

*Application Note:*

The keys are reset as specified in **[JCAPI]** Key class, with the method clearKey(). Any access to a cleared key for ciphering or signing shall throw an exception.

## FCS_COP.1 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 table below:**

| Cryptographic operation | Cryptographic algorithm | TOE supported size | "Standardized" size | List of standards |
|---|---|---|---|---|
| signature, signature's verification, encryption and decryption | DES – TDES with Modes CBC, ECB, and CMAC | 56, 112 or 168 bits | TDES 168 | FIPS PUB 46-3, ANSI X3.92, FIPS PUB 81, ISO/IEC 9797(1999), Data integrity |

| | | | | |
|---|---|---|---|---|
| | | | | **mechanism [FIPS_DES]** |
| **signature, signature's verification, encryption and decryption** | **AES with Modes CBC, ECB, and CMAC** | **from 128 to 256 bits with a step of 64 bits** | **greater than 128 bits** | **FIPS PUB 197 SP800-38B (CMAC)** |
| **signature, signature's verification** | **ECDSA** | **192, 256, 384, 512 and 521 bits** | **Greater than 200 bits** | **ANSI X9.62-1998 [ECDSA] and BRAINPOOL (RFC 5639)** |
| **Key agreement** | **ECDH** | **192 to 521 bits** | **Greater than 200 bits** | **IEEE P1363 [IEEE] and BRAINPOOL (RFC 5639)** |
| **Hash functions** | **SHA-1, SHA-224, SHA-256 and SHA-512** | **SHA-1, SHA-224, SHA-256 and SHA-512** | **SHA-224, SHA-256 and SHA-512** | **Secure Hash Standard, FIPS PUB 180-3** |
| **Checksum** | **16-bit using the hardware co-processor** | **16 bits** | | **ISO3309** |
| **Checksum** | **32-bit in software implementation** | **32 bits** | | **ISO3309** |

.

*Refinement:*

TDES (IC)/IDEMIA has developed the algorithm using HW DES module/TDES encryption and decryption/Triple Data Encryption (TDES)/56/112/168-bits/E-D-E triple- encryption implementation of the Data Encryption Standard, FIPS PUB 46-3, 25 Oct. 1999

SHA /IDEMIA has developed the algorithm/Hash function/SHA-1/No cryptographic key/Secure Hash Standard, Federal Information Processing Standards Publication 180-3, 2008, october

SHA /IDEMIA has developed the algorithm/Hash function/SHA-224/No cryptographic key/Secure Hash Standard, Federal Information Processing Standards Publication 180-3, 2008, october

SHA /IDEMIA has developed the algorithm/Hash function/SHA-256/No cryptographic key/Secure Hash Standard, Federal Information Processing Standards Publication 180-3, 2008, october

SHA /IDEMIA has developed the algorithm/Hash function/SHA-512/No cryptographic key/Secure Hash Standard, Federal Information Processing Standards Publication 180-3, 2008, october

KG /IDEMIA has developed the algorithm using HW PK accelerator/Key Generator//Between 1024 bits to 2048 bits/

AES/IDEMIA has developed the algorithm/Data encryption / decryption//128/192/256 bits/FIPS PUB 197, 2001, November

*Application Note:*

- The TOE shall provide a subset of cryptographic operations defined in **[JCAPI]** (see javacardx.crypto.Cipher and javacardx.security packages).
- For Elliptic curves ECDSA and ECDH, the TOE shall provide a subset of cryptographic operations ECC Brainpool: the Named curves and their domain parameters are introduced in package javacard.security.NamedParameterSpec:
  - Named curves Permanently supported by the TOE as standard curves are:
    - BRAINPOOLP256R1
    - BRAINPOOLP384R1
    - SECP256R1
    - SECP384R1
  - Optionally supported (i.e. to be loaded in personalization)
    - BRAINPOOLP192R1, BRAINPOOLP192T1, BRAINPOOLP224R1, BRAINPOOLP224T1, BRAINPOOLP256T1, BRAINPOOLP320R1, BRAINPOOLP320T1, BRAINPOOLP384T1, BRAINPOOLP512R1, BRAINPOOLP512T1
    - FRP256V1
    - SECP192R1, SECP224R1, SECP521R1

**FDP_RIP.1/ABORT Subset residual information protection**

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

*Application Note:*

The events that provoke the de-allocation of a transient object are described in **[JCRE]**, §5.1.

**FDP_RIP.1/APDU Subset residual information protection**

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

**FDP_RIP.1/GlobalArray Subset residual information protection**

**FDP_RIP.1.1/GlobalArray [Editorially Refined]** The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** *the applet as a result of returning from the process method to* the following objects: **a user Global Array**.

*Application Note:*

An array resource is allocated when a call to the API method JCSystem.makeGlobalArray is performed. The Global Array is created as a transient JCRE Entry Point Object ensuring that reference to it cannot be retained by any application. On return from the method which called JCSystem.makeGlobalArray, the array is no longer available to any applet and is deleted and the memory in use by the array is cleared and reclaimed in the next object deletion cycle.

**FDP_RIP.1/bArray Subset residual information protection**

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

*Application Note:*

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

**FDP_RIP.1/KEYS Subset residual information protection**

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

*Application Note:*

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

| **FDP_RIP.1/TRANSIENT Subset residual information protection** |
|---|

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

*Application Note:*

- The events that provoke the de-allocation of any transient object are described in **[JCRE]**, §5.1.
- The clearing of CLEAR_ON_DESELECT objects is not necessarily performed when the owner of the objects is deselected. In the presence of multiselectable applet instances, CLEAR_ON_DESELECT memory segments may be attached to applets that are active in different logical channels. Multiselectable applet instances within a same package must share the transient memory segment if they are concurrently active (**[JCRE]**, §4.3.

| **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]**.

*Application Note:*

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

**FCS_RNG.1 Random Number Generation**

**FCS_RNG.1.1** The TSF shall provide a **deterministic hybrid** random number generator that implements **CTR_DRBG as defined in NIST SP800-90**.

**FCS_RNG.1.2** The TSF shall provide random numbers that meet **the average Shannon entropy per internal random bit exceeds 0.994**.

#### 7.1.1.2 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.CAP_FILE, 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,
- OP.ARRAY_LENGTH,
- OP.ARRAY_AASTORE.

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

*Application Note:*

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

**FDP_ACF.1/FIREWALL Security attribute based access control**

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

| **Subject/Object** | **Security attributes** |
|---|---|
| **S.CAP_FILE** | **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.CAP_FILE may freely perform 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.CAP_FILE 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.CAP_FILE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT with Context attribute different from the currently active context, 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.CAP_FILE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT with Context attribute different from the currently active context, 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.CAP_FILE may perform OP.CREATE upon O.JAVAOBJECT only if the value of the Sharing parameter is "Standard" or "SIO".**
- **R.JAVA.6 ([JCRE], §6.2.8): S.CAP_FILE may freely perform OP.ARRAY_ACCESS or OP.ARRAY_LENGTH upon any O.JAVAOBJECT whose Sharing attribute has value "global array"**.

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

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

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

- **Any subject with OP.JAVA upon an O.JAVAOBJECT whose LifeTime attribute has value "CLEAR_ON_DESELECT" if O.JAVAOBJECT's Context attribute is not the same as the Selected Applet Context.**
- **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.**
- **S.CAP_FILE performing OP.ARRAY_AASTORE of the reference of an O.JAVAOBJECT whose sharing attribute has value "global array" or "Temporary JCRE entry point".**
- **S.CAP_FILE performing OP.PUTFIELD or OP.PUTSTATIC of the reference of an O.JAVAOBJECT whose sharing attribute has value "global array" or "Temporary JCRE entry point"**

*Application Note:*

FDP_ACF.1.4/FIREWALL:

The following rules related to the firewall are implemented in the product, are out of the scope of the security target.

- R.JAVA.7 ([JCRE], §6.2.8.2): S.CAP_FILE performing OP.ARRAY_T_ASTORE into an array view without ATTR_WRITABLE_VIEW access attribute.
- R.JAVA.8 ([JCRE], §6.2.8.2):S.CAP_FILE performing OP.ARRAY_T_ALOAD into an array view without ATTR_READABLE_VIEW access attribute.

FDP_ACF.1.4/FIREWALL:

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

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

The Sharing attribute defines four categories of objects:

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

- Shareable interface Objects (SIO), which provide a secure mechanism for inter-applet communication,
- JCRE entry points (Temporary or Permanent), who have freely accessible methods but protected fields,
- Global arrays, having both unprotected fields (including components; refer to JavaCardClass discussion above) and methods.
- Array Views, having fields/elements access controlled by access control attributes, ATTR_READABLE_VIEW and ATTR_WRITABLE_VIEW and methods.

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

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

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

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

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

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

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

## FDP_IFC.1/JCVM Subset information flow control

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

*Application Note:*

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

## FDP_IFF.1/JCVM Simple security attributes

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

| 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 "Java Card RE";**
- **other OP.PUT operations are allowed regardless of the Currently Active Context's value**.

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

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

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

*Application Note:*

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

It should be noticed that this policy essentially applies to the execution of bytecode. The Java Card RE itself and possibly some API methods can be granted specific rights or limitations through the FDP_IFF.1.3/JCVM to FDP_IFF.1.5/JCVM elements. The way the Java Card virtual

machine manages the transfer of values on the stack and local variables (returned values, uncaught exceptions) from and to internal registers is implementation-dependent. For instance, a returned reference, depending on the implementation of the stack frame, may transit through an internal register prior to being pushed on the stack of the invoker. The returned bytecode would cause more than one OP.PUT operation under this scheme.

**FDP_RIP.1/OBJECTS Subset residual information protection**

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

*Application Note:*

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

**FMT_MSA.1/JCRE Management of security attributes**

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

*Application Note:*

The modification of the Selected Applet Context should be performed in accordance with the rules given in **[JCRE]**, §4 and **[JCVM]**, §3.4.

**FMT_MSA.1/JCVM Management of security attributes**

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

*Application Note:*

The modification of the Currently Active Context should be performed in accordance with the rules given in **[JCRE]**, §4 and **[JCVM]**, §3.4.

## FMT_MSA.2/FIREWALL_JCVM Secure security attributes

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

*Application Note:*

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

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

## FMT_MSA.3/FIREWALL Static attribute initialisation

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

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

*Application Note:*

FMT_MSA.3.1/FIREWALL

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

FMT_MSA.3.2/FIREWALL

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

**FMT_MSA.3/JCVM Static attribute initialisation**

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

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

**FMT_SMF.1 Specification of Management Functions**

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

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

**FMT_SMR.1 Security roles**

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

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

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

#### 7.1.1.3 Card Security Management

**FAU_ARP.1 Security alarms**

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

- **throw an exception,**
- **lock the card session,**
- **reinitialize the Java Card System and its data,**
- **response with error code to S.CAD**

**,** 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 R[7], §7.6.2)
- violation of the Firewall or JCVM SFPs,
- unavailability of resources,
- array overflow

| **FDP_SDI.2/DATA Stored data integrity monitoring and action** |
|---|

**FDP_SDI.2.1/DATA** The TSF shall monitor user data stored in containers controlled by the TSF for **integrity errors** on all objects, based on the following attributes: **integrityControlledData**.

**FDP_SDI.2.2/DATA** Upon detection of a data integrity error, the TSF shall **increase counter of the Killcard file. If the maximum is reached the killcard is launched**.

*Application Note:*

The following data persistently stored by TOE have the user data attribute "integrityControlledData ":

- PINs (i.e. objects instance of class OwnerPin or subclass of interface PIN)
- Keys (i.e. objects instance of classes implemented the interface Key)
- SecureStores (i.e. objects instance of class SecureStore)
- Packages Java Card
- Patches
- Biometric Data
- Personalization Dump

| **FPR_UNO.1 Unobservability** |
|---|

**FPR_UNO.1.1 [Editorially Refined]** The TSF shall ensure that **any user** is unable to observe the operation **Cardholder authentication** on **D.PIN and D.BIO** by **no user and no subject**.

*Application Note:*

The non-observability of operations on sensitive information such as keys appears as impossible to circumvent in the smart card world. The precise list of operations and objects is left unspecified, but should at least concern secret keys and PIN values when they exist on the card, as well as the cryptographic operations and comparisons performed on them.

**FPT_FLS.1 Failure with preservation of secure state**

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

*Application Note:*

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

**FPT_TDC.1 Inter-TSF basic TSF data consistency**

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

**FPT_TDC.1.2** The TSF shall use

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

when interpreting the TSF data from another trusted IT product.

*Application Note:*

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

#### 7.1.1.4 AID Management

**FIA_ATD.1/AID User attribute definition**

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

- **Package AID,**
- **Applet's version number,**
- **Registered applet AID,**

*Refinement:*

"Individual users" stand for applets.

| **FIA_UID.2/AID User identification before any action** |
|---|

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

*Application Note:*

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

| **FIA_USB.1/AID User-subject binding** |
|---|

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

**FIA_USB.1.2/AID** The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **for each loaded package is associated an unique Package AID**.

**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 initially assigned Package AID is unchangeable**.

*Application Note:*

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

**FMT_MTD.1/JCRE Management of TSF data**

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

*Application Note:*

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

**FMT_MTD.3/JCRE Secure TSF data**

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

### *7.1.2 InstG Security Functional Requirements*

This group consists of the SFRs related to the installation of the applets, which addresses security aspects outside the runtime. The installation of applets is a critical phase, which lies partially out of the Boundary 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 **CAP FILE 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: **CAP FILE 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)**.

*Application Note:*

FDP_ITC.2.1/Installer:

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

FDP_ITC.2.3/Installer:

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

FDP_ITC.2.4/Installer:

- Each package contains a package Version attribute, which is a pair of major and minor version numbers (**[JCVM]**, §4.5). With the AID, it describes the package defined in the CAP file. When an export file is used during preparation of a CAP file, the versions numbers and AIDs indicated in the export file are recorded in the CAP files (**[JCVM]**, §4.5.2): the dependent packages Versions and AIDs attributes allow the retrieval of these identifications. Implementation-dependent checks may occur on a case-by-case basis to indicate that package files are binary compatible. However, package files do have "package Version Numbers" (**[JCVM]**) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.

FDP_ITC.2.5/Installer:

- A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs.
- The intent of this rule is to ensure the binary compatibility of the package with those already on the card (**[JCVM]**, §4.4).
- The installation (the invocation of an applet's install method by the installer) is implementation dependent (**[JCRE]**, §11.2).
- Other rules governing the installation of an applet, that is, its registration to make it SELECTable by giving it a unique AID, are also implementation dependent (see, for example, **[JCRE]**, §11).

**FMT_SMR.1/Installer Security roles**

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

*Application Note:*

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

**FPT_RCV.3/Installer Automated recovery without undue loss**

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

*Refinement:*

There is no maintenance mode on the TOE.

**FPT_RCV.3.2/Installer** For **a failure during load/installation of a package/applet and deletion of a package/applet/object**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

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

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

*Application Note:*

FPT_RCV.3.1/Installer:

- This element is not within the scope of the Java Card specification, which only mandates the behaviour of the Java Card System in good working order. Further details on the "maintenance mode" shall be provided in specific implementations. The following is an except from **[CC2]**, p296: In this maintenance mode normal operation might be impossible or severely restricted, as otherwise insecure situations might occur. Typically, only authorised users should be allowed access to this mode but the real details of who can access this mode is a function of FMT: Security management. If FMT: Security management does not put any controls on who can access this mode, then it may be acceptable to allow any user to restore the system if the TOE enters such a state. However, in practice, this is probably not desirable as the user restoring the system has an opportunity to configure the TOE in such a way as to violate the SFRs.

FPT_RCV.3.2/Installer:

- Should the installer fail during loading/installation of a package/applet, it has to revert to a "consistent and secure state". The Java Card RE has some clean up duties as well; see **[JCRE]**, §11.1.5 for possible scenarios. Precise behaviour is left to implementers. This component shall include among the listed failures the deletion of a package/applet. See (**[JCRE]**, 11.3.4) for possible scenarios. Precise behaviour is left to implementers.
- Other events such as the unexpected tearing of the card, power loss, and so on, are partially handled by the underlying hardware platform (see **[PP0084]**) and, from the TOE's side, by events "that clear transient objects" and transactional features. See FPT_FLS.1.1, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ABORT and FDP_ROL.1/FIREWALL.

FPT_RCV.3.3/Installer:

- The quantification is implementation dependent, but some facts can be recalled here. First, the SCP ensures the atomicity of updates for fields and objects, and a power-failure during a transaction or the normal runtime does not create the loss of otherwise-permanent data, in the sense that memory on a smart card is essentially persistent with this respect (Flash). Data stored on the RAM and subject to such failure is intended to have a limited lifetime anyway (runtime data on the stack, transient objects' contents). According to this, the loss of data within the TSF scope should be limited to the same restrictions of the transaction mechanism.

### 7.1.3 ADELG Security Functional Requirements

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

**FDP_ACC.2/ADEL Complete access control**

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

*Refinement:*

The operations involved in the policy are:

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

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

**FDP_ACF.1/ADEL Security attribute based access control**

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

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

.

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

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

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

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

- **R.JAVA.14 ([JCRE], §11.3.4.2, Applet Instance Deletion): S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if,**
  - **(1) S.ADEL is currently selected,**
  - **(2) there is no instance in the context of O.APPLET that is active in any logical channel and**
  - **(3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE], §8.5) O.JAVAOBJECT is remote reachable.**
- **R.JAVA.15 ([JCRE], §11.3.4.2.1, Multiple Applet Instance Deletion): S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if,**
  - **(1) S.ADEL is currently selected,**
  - **(2) there is no instance of any of the O.APPLET being deleted that is active in any logical channel and**
  - **(3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE], §8.5) O.JAVAOBJECT is remote reachable.**
- **R.JAVA.16 ([JCRE], §11.3.4.3, 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], §11.3.4.4, Applet Package and Contained Instances Deletion): S.ADEL may perform OP.DELETE_PCKG_APPLET upon an O.CODE_PKG only if,**
  - **(1) S.ADEL is currently selected,**
  - **(2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_PKG, which is an instance of a class that belongs to O.CODE_PKG exists on the card,**
  - **(3) there is no package loaded on the card that depends on O.CODE_PKG, and**

- **(4) for every O.APPLET of those being deleted it holds that: (i) there is no instance in the context of O.APPLET that is active in any logical channel and (ii) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted, or ([JCRE], §8.5) O.JAVAOBJECT is remote reachable**.

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

**FDP_ACF.1.4/ADEL** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **any subject but S.ADEL to O.CODE_PKG or O.APPLET for the purpose of deleting them from the card**.

*Application Note:*

FDP_ACF.1.2/ADEL:

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

### FDP_RIP.1/ADEL Subset residual information protection

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

*Application Note:*

Deleted freed resources (both code and data) may be reused, depending on the way they were deleted (logically or physically). Requirements on de-allocation during applet/package deletion are described in **[JCRE]**, §11.3.4.2, §11.3.4.3 and §11.3.4.4.

### FMT_MSA.1/ADEL Management of security attributes

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

**FMT_MSA.3/ADEL Static attribute initialisation**

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

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

**FMT_SMF.1/ADEL Specification of Management Functions**

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

*Application Note:*

The modification of the Active Applets security attribute should be performed in accordance with the rules given in **[JCRE]**, §4.

**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**.

*Application Note:*

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

### *7.1.4 ODELG Security Functional Requirements*

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

**FDP_RIP.1/ODEL Subset residual information protection**

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

*Application Note:*

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

**FPT_FLS.1/ODEL Failure with preservation of secure state**

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

*Application Note:*

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

### *7.1.5 CarG Security Functional Requirements*

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

#### 7.1.5.1 Miscellaneous

**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** The TSF shall be able to relate the **identity** of the originator of the information, and the **application CAP file** of 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**.

*Application Note:*

FCO_NRO.2.1/CM:

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

FCO_NRO.2.3/CM:

- The exact limitations on the evidence of origin are implementation dependent. In most of the implementations, the card manager performs an immediate verification of the origin of the package using an electronic signature mechanism, and no evidence is kept on the card for future verifications.

**FDP_IFC.2/CM Complete information flow control**

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

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

*Application Note:*

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

- The information controlled by the policy is the APDUs exchanged by the subjects through the communication channel linking the card and the CAD. Each of those messages contain part of an application package that is required to be loaded on the card, as well as any control information used by the subjects in the communication protocol.
- When CLFDB is used, the Load File Data Block Hash is associated to the data blocks according section 5.8.3 of **[GP4]**. The hash algorithm used depends on Card Capabilities associated to the SD.

| **FDP_IFF.1/CM Simple security attributes** |
|---|

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

**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 used by the CAD and the card for transmitting a new package, see chapter 9.3.9 [GP1]**.

**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: **the rules describing the communication protocol used by the CAD and the card for transmitting a new package, see chapter 9.3.9 [GP1]**.

*Application Note:*

FDP_IFF.1.1/CM:

- The security attributes used to enforce the CAP FILE LOADING SFP are implementation dependent. More precisely, they depend on the communication protocol enforced between the CAD and the card. For instance, some of the attributes that can be used are: (1) the keys used by the subjects to encrypt/decrypt their messages; (2) the number of pieces the application package has been split into in order to be sent to the card; (3) the ordinal of each piece in the decomposition of the package, etc. See for example Appendix D of **[GP2]**.

FDP_IFF.1.2/CM:

- The precise set of rules to be enforced by the function is implementation dependent. The whole exchange of messages shall verify at least the following two rules: (1) the subject S.INSTALLER shall accept a message only if it comes from the subject S.CAD; (2) the subject S.INSTALLER shall accept an application package only if it has received without modification and in the right order all the APDUs sent by the subject S.CAD.

### FDP_UIT.1/CM Data exchange integrity

**FDP_UIT.1.1/CM** The TSF shall enforce the **CAP FILE LOADING information flow control SFP** to **receive** user data in a manner protected from **deletion, insertion, replay 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 and replay** of some of the pieces of the application sent by the CAD has occurred.

*Application Note:*

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

### FIA_UID.1/CM Timing of identification

**FIA_UID.1.1/CM** The TSF shall allow **Execution of Card Manager** on behalf of the user to be performed before the user is identified.

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

*Application Note:*

The list of TSF-mediated actions is implementation-dependent, but package installation requires the user to be identified. Here by user is meant the one(s) that in the Security Target shall be associated to the role(s) defined in the component FMT_SMR.1/CM.

### FMT_MSA.1/CM Management of security attributes

**FMT_MSA.1.1/CM** The TSF shall enforce the **CAP FILE LOADING information flow control SFP** to restrict the ability to **modify** the security attributes **AS.KEYSET_VERSION, AS.KEYSET_VALUE, Default SELECTED Privileges, AS.CMLIFECYC** to **R.Card_Manager**.

**FMT_MSA.3/CM Static attribute initialisation**

**FMT_MSA.3.1/CM** The TSF shall enforce the **CAP FILE 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 **Card manager** 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: **Modify the following security attributes: AS.KEYSET_VERSION, AS.KEYSET_VALUE, Default SELECTED Privileges, AS.CMLIFECYC**.

**FMT_SMR.1/CM Security roles**

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

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

**FTP_ITC.1/CM Inter-TSF trusted channel**

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

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

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

*Application Note:*

New packages can be installed on the card only on demand of the card issuer.

#### 7.1.5.2 Additional Security Functional Requirements for CM

**FPT_TST.1 TSF testing**

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

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

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

*Application Note:*

Namely, "stored TSF executable code" encompasses the patch and java packages. During startup, the TOE checks the integrity of the patch/java packages. To do so, the related bits should have been set accordingly at the pre-personalisation phase, c.f. AGD_PRE, **[AGD_PRE]** section Lock POST (DO 'DF41').

Other self-tests are described in AGD_PRE, **[AGD_PRE]** section Lock POST (DO 'DF41'). Namely, According to the protocol used Known Answer Test (or POST for Power On Self Tests) checks SHA, RSA, ECDSA either in startup or during 1st use. Those latter tests are configurable.

RNG, CRC, DES and AES set of self-tests can be performed in startup, regarding the configuration.

**FCO_NRO.2/CM_DAP Enforced proof of origin**

**FCO_NRO.2.1/CM_DAP** The TSF shall enforce the generation of evidence of origin for transmitted **Loadfile** at all times.

**FCO_NRO.2.2/CM_DAP** The TSF shall be able to relate the **AS.KEYSET_VALUE** of the originator of the information, and the **CAP file components** of the information to which the evidence applies.

**FCO_NRO.2.3/CM_DAP** The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **during CAP file loading**.

*Application Note:*

This feature included in this st allows an Application Provider to require that their Application code to be loaded on the card shall be checked for integrity and authenticity. The DAP Verification Key is identified by the Key Version Number '73' and the Key Identifier '01'.

See description in §9.2.1 of GlobalPlatform Card Specification for more details [9].

In this implementation, DAPs are generated and verified according the one of the following schemes:

- When RSA module is embedded, the RSA scheme (Variant 1) specified in appendix C.6.1 of [9] is supported. For this scheme, the DAP Verification Key shall be a 1024-bits RSA public key.
- When RSA module is embedded, the RSA scheme (Variant 2) specified in § 5.2 of GlobalPlatform specification Amendment D [45]. For this scheme, the DAP Verification Key shall be a 2048-bits RSA public key. The algorithm is RSASSA-PSS as defined in PKCS#1.
- The AES scheme specified in appendix C.6.1 of [9] is supported. For this scheme, the DAP Verification Key shall be a 128-bits AES key.

**FIA_AFL.1/CM Authentication failure handling**

**FIA_AFL.1.1/CM** The TSF shall detect when **1** unsuccessful authentication attempts occur related to **U.Card_Issuer authentication**.

**FIA_AFL.1.2/CM** When the defined number of unsuccessful authentication attempts has been **met**, the TSF shall **slow down exponentially the next authentication**.

**FIA_UAU.1/CM Timing of authentication**

**FIA_UAU.1.1/CM** The TSF shall allow **Get Data, Initialize Update, Select** on behalf of the user to be performed before the user is authenticated.

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

**FIA_UAU.4/CardIssuer Single-use authentication mechanisms**

**FIA_UAU.4.1/CardIssuer** The TSF shall prevent reuse of authentication data related to **Authentication Mechanism based on Triple-DES and/or AES**.

*Application Note:*

The authentication mechanism, used to open a secure channel communication with the card issuer, use a challenge freshly and randomly generated by the TOE in order to prevent reuse of a response generated by a terminal in a successful authentication attempt.

### FIA_UAU.7/CardIssuer Protected authentication feedback

**FIA_UAU.7.1/CardIssuer** The TSF shall provide only **the result of the authentication (NOK), the key set version, Secure channel identifier and the card random and the card cryptogram** to the user while the authentication is in progress.

### FPR_UNO.1/Key_CM Unobservability

**FPR_UNO.1.1/Key_CM** The TSF shall ensure that **all subjects** are unable to observe the operation **OP.IMPORT_KEY** on **Key** by **D.JCS_KEYS**.

### FPT_TDC.1/CM Inter-TSF basic TSF data consistency

**FPT_TDC.1.1/CM** The TSF shall provide the capability to consistently interpret **AS.KEYSET_VALUE** when shared between the TSF and another trusted IT product.

**FPT_TDC.1.2/CM** The TSF shall use **the rules defined in the GP [GP1] section 11.8** when interpreting the TSF data from another trusted IT product.

### FMT_SMR.2/CM Restrictions on security roles

**FMT_SMR.2.1/CM** The TSF shall maintain the roles: **see below**.

**FMT_SMR.2.2/CM** The TSF shall be able to associate users with roles.

**FMT_SMR.2.3/CM** The TSF shall ensure that the conditions **see details below:**

| Roles | Condition for this role |
|---|---|
| R.personaliser | Successful authentication (Card Issuer) using a key set of the Card Manager or Security Domain associates with CM life cycle phase from OP_READY to SECURED |
| R.Card_Manager | Successful authentication (of Card Issuer) using its key set, with CM life cycle phase from OP_READY to SECURED |
| R.Security_Domain | Successful authentication (of application provider) using its key set, with CM life cycle phase different from locked |
| R.Use_API | Successful identification (of Applet), with Applet life cycle phase after SELECTABLE |
| R.Applet_privilege | have the privilege to modify CM life cycle, ATR, and also Global Pin |

**FCS_COP.1/CM Cryptographic operation**

**FCS_COP.1.1/CM** 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:

| Cryptographic operation | Algorithm | Key length | Standard |
|---|---|---|---|
| TOE authentication key ISK/KMC | SCP02 | 112 bits | GP 2.3 |
| TOE authentication key ISK/KMC | SCP03 | 128/192/256 bits | GP 2.3 |
| SCP02 - signature, verification of signature, encryption and decryption (KEK (key encryption key) for sensitive objects such as PIN, keys ... is mandatory) | TDES | 112 bits | SCP02 – GP 2.3 |
| SCP03 - signature, verification of signature, encryption and decryption | AES | 128/192/256 bits | SCP03 – GP 2.3 |
| Secure messaging (GP) - encryption and decryption | Triple-DES in CBC mode | 112 bits | FIPS PUB 46-3 (ANSI X3.92) [FIPS1] |
| Secure messaging (GP) - encryption and decryption | AES in CBC mode | 128, 192 and 256 bits | FIPS PUB 197 [FIPS_AES] |
| Secure messaging (GP) - key establishment protocol for the PIV | AES CBC 8-bytes MAC | bits 128 or 256 bits | NIST SP 800-73-4 |
| Decryption of Ciphered Load File Data Blocks (GP) | TDES with CBC mode | 128 bits | FIPS PUB 46-3 (ANSI X3.92) [FIPS1] |
| Decryption of Ciphered Load File Data Blocks (GP) | AES with CBC mode | 128, 192 and 256 bits | FIPS PUB 197 [FIPS_AES] |

.

### 7.1.5.3 Additional Security Functional Requirements for Resident application

## FDP_ACC.2/PP Complete access control

**FDP_ACC.2.1/PP [Editorially Refined]** The TSF shall enforce the **Access Control** on **see application note below** and all operations among subjects and objects covered by the SFP.

| | |
|---|---|
| Access Control | |
| Administration Access Control | S.Resident application and for all objects |
| Patch Loading Access Control and ISK key | S.TOE and for all objects |
| Dump Perso Access Control and DSK | S.Resident application and for all objects |

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

*Application Note:*

This SFR enforces the access control for the patch, dump perso, locks loading and the ISK loading.

## FDP_ACF.1/PP Security attribute based access control

**FDP_ACF.1.1/PP** The TSF shall enforce the **Access Control on See below** to objects based on the following:

| **Subject** | **Attribute** |
|---|---|
| **R.Prepersonaliser** | **AS_AUTH_MSK_STATUS** |

.

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

- **R.Prepersonaliser is allowed to load a patch if:**
  - **correctly encrypted with the LSK key**
  - **the memory area to be modified is genuine**.

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

**FDP_ACF.1.4/PP** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none**.

*Application Note:*

The dedicated key will be LSK in case of secure update and JSK in case of patch loading. The patch loading before use phase is in mode 1 using JSK. The patch loading in user Phase is in mode 2 using a diversification of JSK for encryption and computing an additional MAC of the patch.

### FDP_UCT.1/PP Basic data exchange confidentiality

**FDP_UCT.1.1/PP** The TSF shall enforce the **Administration access control and Patch access control** to **receive** user data in a manner protected from unauthorised disclosure.

*Application Note:*

For the Administration access control, the MSK is used to cipher the data transmitted (ISK). For the Patch and Locks loading access control, the LSK is used to cipher the data transmitted. For Perso dump, the DSK is used to cipher the data transmitted.

### FDP_ITC.1/PP Import of user data without security attributes

**FDP_ITC.1.1/PP** The TSF shall enforce the **Administration access control and Patch access control** when importing user data, controlled under the SFP, from outside of the TOE.

**FDP_ITC.1.2/PP** The TSF shall ignore any security attributes associated with the user data when imported from outside the TOE.

**FDP_ITC.1.3/PP** The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TOE: **none**.

### FIA_AFL.1/PP Authentication failure handling

**FIA_AFL.1.1/PP** The TSF shall detect when **3** unsuccessful authentication attempts occur related to **U.Card_manufacturer authentication**.

**FIA_AFL.1.2/PP** When the defined number of unsuccessful authentication attempts has been **met**, the TSF shall **always return an error**.

**FIA_UAU.1/PP Timing of authentication**

**FIA_UAU.1.1/PP** The TSF shall allow **INITIALIZE AUTHENTICATION PROCESS, GET DATA, MANAGE CHANNEL, SELECT APPLET** on behalf of the user to be performed before the user is authenticated.

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

**FIA_UID.1/PP Timing of identification**

**FIA_UID.1.1/PP** The TSF shall allow **INITIALIZE AUTHENTICATION PROCESS, GET DATA, MANAGE CHANNEL, and SELECT APPLET** on behalf of the user to be performed before the user is identified.

**FIA_UID.1.2/PP** 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/PP Management of security attributes**

**FMT_MSA.1.1/PP** The TSF shall enforce the **Administration access control** to restrict the ability to **modify** the security attributes **AS.AUTH_MSK_STATUS, MSK Key value, MSK Key counter** to **R.Prepersonaliser**.

**FMT_SMF.1/PP Specification of Management Functions**

**FMT_SMF.1.1/PP** The TSF shall be capable of performing the following management functions: **modify the MSK keys of the Card Manufacturer in Prepersonalisation phase after a successful authentication with the MSK**.

**FIA_UAU.4/CardManu Single-use authentication mechanisms**

**FIA_UAU.4.1/CardManu** The TSF shall prevent reuse of authentication data related to **Authentication Mechanism based on Triple-DES and/or AES**.

### FIA_UAU.7/CardManu Protected authentication feedback

**FIA_UAU.7.1/CardManu** The TSF shall provide only **the result of the authentication (NOK) and the random** to the user while the authentication is in progress.

### FMT_MOF.1/PP Management of security functions behaviour

**FMT_MOF.1.1/PP [Editorially Refined]** The TSF shall restrict the ability to **see below** the functions **see below** to

| | Functions | Role |
|---|---|---|
| **Disable** | **INITIALIZE AUTHENTICATION PROCESS, EXTERNAL AUTHENTICATE, INSTALL, UPDATE SECURE, LOAD APPLET, GET DATA** | **R.Prepersonaliser** |
| **Modify the behaviour of** | **Self-tests described in FPT_TST.1** | **R.Prepersonaliser** |

.

*Application Note:*

The first operation ensures the irreversible locking of the patch and locks loading features once in OP_READY, after pre production state. Once in OP_READY state, those APDU cannot be used.

The second operation described the product configuration regarding self tests, as described in AGD_PRE, chapter 8 **[AGD_PRE]**.

The last operation permits the loading of patch and locks during phase 5.

### FMT_SMR.2/PP Restrictions on security roles

**FMT_SMR.2.1/PP** The TSF shall maintain the roles: **R.Prepersonaliser**.

**FMT_SMR.2.2/PP** The TSF shall be able to associate users with roles.

**FMT_SMR.2.3/PP** The TSF shall ensure that the conditions **see refinement below** are satisfied.

*Refinement:*

| Roles | Condition for this role |
|---|---|
| R.Prepersonaliser | Successful authentication (of Card Manufacturer) using MSK and card still in prepersonalisation state, in phase 4-5. |

## FMT_MSA.3/PP Static attribute initialisation

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

**FMT_MSA.3.2/PP** 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.

## FCS_COP.1/PP Cryptographic operation

**FCS_COP.1.1/PP** 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:

| Cryptographic operation | Algorithm | Key length | Standard |
|---|---|---|---|
| Decryption (MSK) | DES | 112 bits | FIPS-PUB 46-3 (ANSI X3.92), FIPS PUB 81 or ISO/IEC 9797, Data integrity mechanism |
| Card Manufacturer authentication (MSK) | DES | 112 bits | FIPS PUB 197 |
| Card Manufacturer authentication (MSK) | AES | 128, 192 and 256 bits | FIPS-PUB 46-3 (ANSI X3.92), FIPS PUB 81 or ISO/IEC 9797, Data integrity mechanism |
| Decryption of locks ciphered with LSK | AES | 128 bits | FIPS-PUB 46-3 (ANSI X3.92), FIPS PUB 81 or ISO/IEC 9797 |
| Decryption of patch ciphered with JSK | AES | 128 bits | FIPS-PUB 46-3 (ANSI X3.92), FIPS PUB 81 or ISO/IEC 9797 |
| Decryption of patch ciphered with diversified JSK | AES mode CBC and MAC | 128 bits | FIPS-PUB 46-3 (ANSI X3.92), FIPS PUB 81 or ISO/IEC 9797, FIPS PUB 197 SP800-38B (CMAC) |
| Encryption of Perso Dump with DSK | AES | 128 bits | FIPS-PUB 46-3 (ANSI X3.92), FIPS PUB 81 or ISO/IEC 9797 |
| Decryption of Perso Dump ciphered with DSK | AES | 128 bits | FIPS-PUB 46-3 (ANSI X3.92), FIPS PUB 81 or ISO/IEC 9797 |
| ISK MAC verification | DES | 112 bits | FIPS PUB 197 |

.

## FCS_CKM.4/PP Cryptographic key destruction

**FCS_CKM.4.1/PP** The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **Key is set to NULL** that meets the following: **no**.

*Application Note:*

In phase 5, reaching OP_READY state, this SFR ensures the secure erasing of the LSK and MSK keys.

In phases 3-4, MSK is diversified during the first command, and then replaced by the new value generated.

## FDP_UIT.1/PP Data exchange integrity

**FDP_UIT.1.1/PP** The TSF shall enforce the **Patch, Administration loading access control SFP** to **receive** user data in a manner protected from **modification** errors.

**FDP_UIT.1.2/PP [Editorially Refined]** The TSF shall be able to determine on receipt of user data, whether **modification of some of the pieces of the application sent by the TOE developer and Card Manufacturer** has occurred.

*Application Note:*

Modification errors should be understood as modification, substitution, unrecoverable ordering change of data and any other integrity error that may cause the patch to be installed on the card to be different from the one sent by the TOE Developer. The locks loading is performed by the TOE Prepersonaliser via the command UPDATE SECURE described in FCS_COP.1/PP. The ISK loading is performed by the Card Manufacturer via the command PUT KEY, its integrity is ensured by a MAC, described in FCS_COP.1/PP.The Dump perso loaded is protected by a checksum comparison.

**FAU_STG.2 Guarantees of audit data availability**

**FAU_STG.2.1** The TSF shall protect the stored audit records in the audit trail from unauthorised deletion.

**FAU_STG.2.2** The TSF shall be able to **prevent** unauthorised modifications to the stored audit records in the audit trail.

**FAU_STG.2.3** The TSF shall ensure that **Patch code identification** stored audit records will be maintained when the following conditions occur: **audit storage exhaustion, failure and attack**

*Application Note:*

Patch loading steps are:

1. Loading CAP file containing the code in EEPROM (install and load commands)
2. Decryption with LSK key
3. Check the signature (SHA256) of the area to be patched if signature is OK
4. Write the new code (patch itself)
5. Computation of the new SHA signature
6. Deletion of the CAP file stored temporary in EEPROM (DELETE command)

Information on the Patch code (unique identifier) is directly retrieved by GET DATA command.

#### 7.1.5.4 Additional Security Functional Requirements for SmartCard Platform

**FPT_PHP.3/SCP Resistance to physical attack**

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

*Application Note:*

The physical manipulation and physical probing include: changing operational conditions every times: the frequency of the external clock, power supply, and temperature

**FPT_RCV.4/SCP Function recovery**

**FPT_RCV.4.1/SCP** The TSF shall ensure that **reading from and writing to static and objects' fields interrupted by power loss** have the property that the function either

completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

| **FRU_FLT.1/SCP Degraded fault tolerance** |
|---|

**FRU_FLT.1.1/SCP** The TSF shall ensure the operation of **Fault tolerance** when the following failures occur: **failure of flash**.

*Application Note:*

The TOE implements a mechanism to detect a problem of Flash. During the life of the TOE, the Transaction area reduces its size to skip damaged FLASH bytes. During the writing or erasing operations, up to 3 maximum attempts to get successful programming are done.

Otherwise the EXCEPTION_EEPROM_ERROR is raised.

| **FPR_UNO.1/USE_KEY Unobservability** |
|---|

**FPR_UNO.1.1/USE_KEY** The TSF shall ensure that **all subjects** are unable to observe the operation **use** on **key** by **D.JCS_KEYS and APP_KEYs**.

#### 7.1.5.5 Additional Security Functional Requirements for the applets

| **FIA_AFL.1/PIN Authentication failure handling** |
|---|

**FIA_AFL.1.1/PIN** The TSF shall detect when **an administrator configurable positive integer within from 1 to 127 for OwnerPIN** unsuccessful authentication attempts occur related to **any user authentication using a PIN**.

**FIA_AFL.1.2/PIN** When the defined number of unsuccessful authentication attempts has been **met**, the TSF shall **block the PIN**.

**FMT_MTD.2/GP_PIN Management of limits on TSF data**

**FMT_MTD.2.1/GP_PIN** The TSF shall restrict the specification of the limits for **D.NB_REMAINTRYGLB, GlobalPIN** to **R.Card_Manager**.

**FMT_MTD.2.2/GP_PIN** The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: **block D.PIN**

| | |
|---|---|
| **R.Card_Manager** | **Entity after Successful authentication (of Card Issuer) using its key set, with CM life cycle phase from OP_READY to SECURED** |

.

**FMT_MTD.1/PIN Management of TSF data**

**FMT_MTD.1.1/PIN** The TSF shall restrict the ability to **change_default, query and modify** the **OwnerPIN** to **applet itself**.

**FIA_AFL.1/GP_PIN Authentication failure handling**

**FIA_AFL.1.1/GP_PIN** The TSF shall detect when **an administrator configurable positive integer within 3 to 15** unsuccessful authentication attempts occur related to **any user authentication using a Global PIN**.

**FIA_AFL.1.2/GP_PIN** When the defined number of unsuccessful authentication attempts has been **met**, the TSF shall **block the Global PIN**.

#### 7.1.5.6 Additional Security Functional Requirements for Runtime Verification

**Stack Control**

**FDP_ACC.2/RV_Stack Complete access control**

**FDP_ACC.2.1/RV_Stack** The TSF shall enforce the **Stack Access Control SFP** on **S.STACK** and all operations among subjects and objects covered by the SFP.

*Refinement:*

The operations involved in the policy are:

- OP.OPERAND_STACK_ACCESS
- OP.LOCAL_STACK_ACCESS

**FDP_ACC.2.2/RV_Stack** 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/RV_Stack Security attribute based access control**

**FDP_ACF.1.1/RV_Stack** The TSF shall enforce the **Stack Access Control** to objects based on the following:

| Subject/Object | Security attributes |
|---|---|
| S.APPLET | Active Applets, Applet Selection Status |
| S.STACK | Stack Pointer |
| S.JCVM | Current Frame Context |

.

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

- **An Active Applet selected may freely perform OP.LOCAL_STACK_ACCESS upon stack pointer only if the index of the local variable accessed matches the Current Frame Context attribute**
- **An Active Applet selected may freely perform OP.OPERAND_STACK_ACCESS upon Stack Pointer only if the attribute Stack Pointer matches the attribute Current Frame Context of S.JCVM**.

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

**FDP_ACF.1.4/RV_Stack** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none**.

*Application Note:*

Any bytecode accessing a local variable has an index in parameter (byte or short). The first rule aims at verifying that this index is always positive and inferior to the numbers of local variables defined for this stack frame. Then the local variable slot is accessed using the index that is relative to the base of local variables for this stack frame.

Any bytecode accessing the operand stack for push or pop operations is under the control of rule 2. The second rule aims at verifying that the stack pointer is always in the range defined by the base-of-stack and top-of-stack values defined for this stack frame.

The frame context attribute is made of the following elements:

- number-of-local variables and base-of-local-variable
- base-of-stack and top-of-stack

The policies defined in this SFR are enforced dynamically, each time an operation is performed. Nevertheless, those verifications may be redundant with the ones made statically by the off-card verifier, during the applet verification stage.

**FMT_MSA.1/RV_Stack Management of security attributes**

**FMT_MSA.1.1/RV_Stack** The TSF shall enforce the **Stack Access Control SFP** to restrict the ability to **modify** the security attributes **Current Frame Context and Stack Pointer** to **the Java Card VM (S.JCVM)**.

**FMT_MSA.2/RV_Stack Secure security attributes**

**FMT_MSA.2.1/RV_Stack** The TSF shall ensure that only secure values are accepted for **Current Frame Context and Stack Pointer**.

**FMT_MSA.3/RV_Stack Static attribute initialisation**

**FMT_MSA.3.1/RV_Stack** The TSF shall enforce the **Stack Access Control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

**FMT_MSA.3.2/RV_Stack** The TSF shall allow the **any role** to specify alternative initial values to override the default values when an object or information is created.

**FMT_SMF.1/RV_Stack Specification of Management Functions**

**FMT_SMF.1.1/RV_Stack** The TSF shall be capable of performing the following management functions: **Modify the Current Frame Context and modify the Stack Pointer**.

*Application Note:*

The frame context attribute is modified on method invocation. In that case, the previous context attribute is saved on the stack. It will be restored on return of the invoked method.

**Heap Access**

## FDP_ACC.2/RV_Heap Complete access control

**FDP_ACC.2.1/RV_Heap** The TSF shall enforce the **Heap Access Control SFP** on **O.CODE_PKG, O.JAVAOBJECT, S.JCVM, S.APPLET** and all operations among subjects and objects covered by the SFP.

*Refinement:*

The operations involved in the policy are:

- OP.ARRAY_ACCESS
- OP.INSTANCE_FIELD
- OP.STATIC_FIELD
- OP.FLOW

**FDP_ACC.2.2/RV_Heap** 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/RV_Heap Security attribute based access control

**FDP_ACF.1.1/RV_Heap** The TSF shall enforce the **Heap Access Control SFP** to objects based on the following:

| Subject/Object | Security attributes |
|---|---|
| **O.CODE_PKG** | **Package Boundary** |
| **O.JAVAOBJECT** | **Object Boundary** |
| **S.JCVM** | **Program Counter** |
| **S.APPLET** | **Active Applets, Applet Selection Status** |

.

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

- **S.APPLET may freely perform OP.ARRAY_ACCESS and OP.INSTANCE_FIELD upon any O.JAVAOBJECT if the array cell index or the instance field index match the object boundary attribute of O.JAVAOBJECT**
- **S.APPLET may freely perform OP.STATIC_FIELD upon any O.CODE_PKG if the static field index matches the Package Boundary attribute of O.CODE_PKG.**
- **S.APPLET may freely perform OP.FLOW upon O.CODE_PKG if the Program Counter attribute of S.JCVM matches the Package Boundary attribute of O.CODE_PKG**.

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

**FDP_ACF.1.4/RV_Heap** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none**.

*Application Note:*

The upper and lower boundaries of any object allocated on the heap are registered (Object Boundary Attribute). Each time an object is accessed, the first rule verifies that the accessed NVM location is comprised between those two boundaries.

The second rule aims at verifying that when a static field is accessed, the index of this field is positive and inferior to the number of static fields of this package (part of Package Boundary attribute).

The third rule aims at verifying that when a change of execution flow occurs, the computed value for the newly computed value for the Program Counter is comprised within the boundaries defined for this package (part of Package Boundary Attribute). This rule does not concern invocation bytecode.

The policies defined in this SFR are enforced dynamically, each time an operation is performed. Nevertheless, those verifications may be redundant with the ones made statically by the off-card verifier, during the applet verification stage.

**FMT_MSA.1/RV_Heap Management of security attributes**

**FMT_MSA.1.1/RV_Heap** The TSF shall enforce the **Heap Access Control SFP** to restrict the ability to **modify** the security attributes **Package Boundary, Object Boundary and Program Counter** to **S.JCVM**.

**FMT_MSA.2/RV_Heap Secure security attributes**

**FMT_MSA.2.1/RV_Heap** The TSF shall ensure that only secure values are accepted for **Package Boundary, Object Boundary and Program Counter**.

**FMT_MSA.3/RV_Heap Static attribute initialisation**

**FMT_MSA.3.1/RV_Heap** The TSF shall enforce the **Heap Access Control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

**FMT_MSA.3.2/RV_Heap** The TSF shall allow the **no role** to specify alternative initial values to override the default values when an object or information is created.

## FMT_SMF.1/RV_Heap Specification of Management Functions

**FMT_SMF.1.1/RV_Heap** The TSF shall be capable of performing the following management functions: **to modify the Program Counter attribute**.

**Transient Control**

## FDP_ACC.2/RV_Transient Complete access control

**FDP_ACC.2.1/RV_Transient** The TSF shall enforce the **Transient Access Control SFP** on **S.APPLET, S.JCVM and O.JAVAOBJECT** and all operations among subjects and objects covered by the SFP.

*Refinement:*

The operation involved in the policy is:

- OP.ARRAY_ACCESS

**FDP_ACC.2.2/RV_Transient** 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/RV_Transient Security attribute based access control

**FDP_ACF.1.1/RV_Transient** The TSF shall enforce the **Transient Access Control SFP** to objects based on the following:

| Subject/Object | Security Attributes |
|---|---|
| S.APPLET | Active Applets, Applet Selection Status |
| S.JCVM | COR Context, COD Context |
| O.JAVAOBJECT | LifeTime |

.

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

- **S.APPLET may freely perform OP.ARRAY_ACCESS on O.JAVAOBJECT whose LifeTime attribute has value "CLEAR_ON_RESET" only if the targeted volatile memory space matches the COR Context attribute of S.JCVM**
- **S.APPLET may freely perform OP.ARRAY_ACCESS on O.JAVAOBJECT whose LifeTime attribute has value "CLEAR_ON_DESELECT" only if the targeted volatile memory space matches the COD Context attribute of S.JCVM**.

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

**FDP_ACF.1.4/RV_Transient** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **none**.

*Application Note:*

Each time an applet accesses a Clear On Reset (resp. Clear On Deselect) transient, these rules verify that the accessed RAM area is in the range of the Clear On Reset transients space (resp. Clear On Deselect) allocated for all the transients created by the applets of this package.

The COR context attribute represents the lower and upper limits for the Clear On Reset transient space of the active applet package. The COD context attribute represents the lower and upper limits for the Clear On Deselect transient space of the currently selected applet package.

The policies defined in this SFR are enforced dynamically, each time an operation is performed. Nevertheless, those verifications may be redundant with the ones made statically by the off-card verifier, during the applet verification stage.

**FMT_MSA.1/RV_Transient Management of security attributes**

**FMT_MSA.1.1/RV_Transient** The TSF shall enforce the **Transient Access Control SFP** to restrict the ability to **modify** the security attributes **the security attributes COR Context and COD Context** to **Java Card VM (S.JCVM)**.

**FMT_MSA.2/RV_Transient Secure security attributes**

**FMT_MSA.2.1/RV_Transient** The TSF shall ensure that only secure values are accepted for **COR Context and COD Context Security attributes of the Transient Access Control SFP**.

**FMT_MSA.3/RV_Transient Static attribute initialisation**

**FMT_MSA.3.1/RV_Transient** The TSF shall enforce the **Transient Access Control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

**FMT_MSA.3.2/RV_Transient** The TSF shall allow the **no role** to specify alternative initial values to override the default values when an object or information is created.

**FMT_SMF.1/RV_Transient Specification of Management Functions**

**FMT_SMF.1.1/RV_Transient** The TSF shall be capable of performing the following management functions: **modify the COR Context and COD Context Security Attributes**.

### *7.1.6 JBox Security Requirements*

**FDP_ACC.2/JBox Complete access control**

**FDP_ACC.2.1/JBox [Editorially Refined]** The TSF shall enforce the **JBox access control SFP** on **S.TPL, O.TPL_Content, O.Platform_Content, O.JBox_Interface and O.Platform_Interface** and all operations among subjects and objects covered by the SFP. The operations involved in the policy are:

- OP.TPL.ACCESS
- OP.JBox ACCESS_INTERFACE

**FDP_ACC.2.2/JBox** 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/JBox Security attribute based access control**

**FDP_ACF.1.1/JBox** The TSF shall enforce the **JBox access control SFP** to objects based on the following: **S.TPL, O.TPL_Content, O.Platform_Content, O.JBox_Interface, O.Platform_Interface and the MPU/MMU configuration**.

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

- **Code assigned to S.TPL shall only be executed when the active context is the TPL_Context (ie. the TPL context).**
- **Code assigned to S.Platform shall only be excuted when the active context is Platform_Context.**
- **S.TPL Code shall only to able to perform OP.NATIVE_ACCESS to O.TPL_Content. The NVM and the RAM which belongs to O.TPL_Content is controlled by the MMU/MPU configuration**.

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

- **S.TPL Code shall be able to perform OP.NATIVE_INTERFACE_CALL to O.JBox_Interface. This causes a context switch to the Platform Context; context is switched back to TPL_Context on return from this call**.

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

- **S.TPL Code shall not be able to perform OP.NATIVE_ACCESS to O.Platform_Content.**
- **S.TPL Code shall not be able to perform OP.NATIVE _INTERFACE_CALL to O.Platform_Interface.**
- **S.Platform Code shall not be able to perform OP.NATIVE_ACCESS to O.TPL_Content**.

**FMT_MSA.1/JBox Management of security attributes**

**FMT_MSA.1.1/JBox** The TSF shall enforce the **JBox access control SFP** to restrict the ability to **modify** the security attributes **MPU/MMU configuration** to **JCRE**.

**FMT_MSA.3/JBox Static attribute initialisation**

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

**FMT_MSA.3.2/JBox** 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/JBox Specification of Management Functions**

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

- **Switch to the TPL context:**
  - **Change the value in the MPU/MMU configuration to assign RAM to the JBox**
  - **Change the value in the MPU/MMU configuration to assign NVM to the JBox**
- **Switch to the platform context.**
  - **Change the value in the MPU/MMU configuration to assign RAM to the platform.**
  - **Change the value in the MPU/MMU configuration to assign NVM to the platform**.

### *7.1.7 Additional Security Requirements - FLEXICODE package*

#### 7.1.7.1 Additional Security Functionnal Requirements for API - DBI - Module

**FMT_MTD.1/Activate_DBI Management of TSF data**

**FMT_MTD.1.1/Activate_DBI** The TSF shall restrict the ability to **digitally blur** the **image in dedicated applet file** to **the Personalization Agent**.

**FMT_MTD.1/Deactivate_DBI Management of TSF data**

**FMT_MTD.1.1/Deactivate_DBI** The TSF shall restrict the ability to **remove the blurring on** the **digital images** to **the terminal whose name is set by the personalisation agent.**.

#### 7.1.7.2 Miscellaneous

**FMT_MOF.1/FlxC Management of security functions behaviour**

**FMT_MOF.1.1/FlxC** The TSF shall restrict the ability to **enable** the functions **FLEXICODE management by chossing module to delete** to **R.Personaliser and R.Prepersonaliser**.

*Application Note:*

The command PUT DATA is used to delete modules

**FMT_SMF.1/FlxC Specification of Management Functions**

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

- **Securily delete modules (part of code) after the user has choosen which module can be deleted**
- **Manages the List of removable Modules (bits indicators) for module identification**.

**FDP_ACC.2/FlxC Complete access control**

**FDP_ACC.2.1/FlxC [Editorially Refined]** The TSF shall enforce the **Access Control** on **see application note below** and all operations among subjects and objects covered by the SFP.

| | |
|---|---|
| Access Control | |
| Administration Access Control | S.Resident application and for all objects |

| | |
|---|---|
| FLEXICODE Access Control | S.TOE and for all objects |

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

*Application Note:*

This SFR enforces the FLEXICODE Access Control for the modules that can be deleted.

**FDP_ACF.1/FlxC Security attribute based access control**

**FDP_ACF.1.1/FlxC** The TSF shall enforce the **FLEXICODE Access Control on See below** to objects based on the following:

| **Subject** | **Attribute** |
|---|---|
| **S.TOE** | **Modules** |

.

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

- **R.Prepersonaliser and R.Prepersonaliser are allowed to delete a Module:**
  - **the Module is well identified as a removable module**
  - **the Module is independant of the rest of the code**
  - **None Cap File is identified as needing the module**
  - **the memory area to be modified is genuine**.

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

**FDP_ACF.1.4/FlxC** The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **Except the identified modules dedicated to FLEXICODE, All other source code parts and associated objects can't be deleted with FLEXICODE mechanism**.

*Application Note:*

The dedicated key will be LSK in case of secure update and JSK in case of patch loading. The patch loading before use phase is in mode 1 using JSK. The patch loading in user Phase is in mode 2 using a diversification of JSK for encryption and computing an additional MAC of the patch.

### FMT_MSA.1/FlxC Management of security attributes

**FMT_MSA.1.1/FlxC** The TSF shall enforce the **FLEXICODE Access control SFP** to restrict the ability to **change_default** the security attributes **List of removable Modules** to **S.Resident Application**.

### FMT_MSA.2/FlxC Secure security attributes

**FMT_MSA.2.1/FlxC** The TSF shall ensure that only secure values are accepted for **List of removable Modules**.

*Application Note:*

Only the allowed modules can be removed to free memory space.

### FMT_MSA.3/FlxC Static attribute initialisation

**FMT_MSA.3.1/FlxC** The TSF shall enforce the **FLEXICODE Access Control SFP** to provide **permissive** default values for security attributes that are used to enforce the SFP.

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

*Application Note:*

List of removable Modules allows the TSF to remove corresponding module

### FMT_MSA.4/FlxC Security attribute value inheritance

**FMT_MSA.4.1/FlxC** The TSF shall use the following rules to set the value of security attributes: **When a module is effectively deleted the List of removable Modules must be updated**

### FPT_FLS.1/FlxC Failure with preservation of secure state

**FPT_FLS.1.1/FlxC** The TSF shall preserve a secure state when the following types of failures occur: **The module deletion is disrupted during FLEXICODE mechanism**.

**FDP_RIP.1/FlxC Subset residual information protection**

**FDP_RIP.1.1/FlxC** 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: **Modules: the entire source code corresponding to the part of code to delete**.

### *7.1.8 Additional Security Requirements Monotonic Counters package*

**FDP_SDI.2/MONOTONIC_COUNTER Stored data integrity monitoring and action**

**FDP_SDI.2.1/MONOTONIC_COUNTER** The TSF shall monitor user data stored in containers controlled by the TSF for **integrity errors** on all objects, based on the following attributes: **stored user data i.e. the counter value in the MonotonicCounter object**.

**FDP_SDI.2.2/MONOTONIC_COUNTER** Upon detection of a data integrity error, the TSF shall **throw an exception**.

*Application Note:*

This requirement applies to MonotonicCounter objects created by the getInstance() method of the javacardx.security.util.MonotonicCounter class.

### *7.1.9 Additional Security Functional Requirements for Sensitive Array package*

**FDP_SDI.2/ARRAY Stored data integrity monitoring and action**

**FDP_SDI.2.1/ARRAY** 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 data stored in arrays created by the makeIntegritySensitiveArray() method of the javacard.framework.SensitiveArrays class**.

**FDP_SDI.2.2/ARRAY** Upon detection of a data integrity error, the TSF shall **throw an exception**.

### *7.1.10 PACE package Security Functional Requirements*

The SFRs for PACE Module are refined from ICAO-PACE PP **[PP0068]** in a more generic form in order to be applicable to any application requiring PACE protocol. The MRTD is kept as application exemple. The suffix PACE is added to the SFRs since PP SFRs.

*Please refer to **[PP0068]** for all definitions and application notes.*

#### 7.1.10.1 Class FCS Cryptographic Support

## FCS_CKM.1/DH_PACE Cryptographic key generation

**FCS_CKM.1.1/DH_PACE** The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **ECDH compliant to [TR_03111]** and specified cryptographic key sizes **192 to 512 bit** that meet the following: **[ICAO-TR]**.

*Application Note:*

The shared secret value K is used for deriving the AES or DES session keys for message encryption and message authentication according to [48] for the TSF required by FCS_COP.1/PACE_ENC and FCS_COP.1/PACE_MAC. FCS_CKM.1/DH_PACE implicitly contains the requirements for the hashing functions used for key derivation by demanding compliance to **[ICAO-TR]**.

## FCS_COP.1/PACE_ENC Cryptographic operation

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

| Cryptographic Operations | Algorithms | Key sizes | Norms |
|---|---|---|---|
| secure messaging-encryption and decryption | AES in CBC mode | 128, 192 and 256 bits | [ICAO-TR] |
| secure messaging-encryption and decryption | TDES in CBC mode | 112 bits | [ICAO-TR] |

.

*Application Note:*

This SFR requires the TOE to implement the cryptographic primitive AES or 3DES for secure messaging with encryption of transmitted data and encrypting the nonce in the first step of PACE. The secure messing is implemented by FCS_COP.1.1/CM. The related session keys are agreed between the TOE and the terminal as part of the PACE protocol according to the FCS_CKM.1/DH_PACE (PACE-KEnc).

## FCS_COP.1/PACE_MAC Cryptographic operation

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

| Cryptographic Operations | Algorithms | Key sizes | Norms |
|---|---|---|---|
| secure messaging - message authentication code | AES CMAC | 128, 192 and 256 bits | [ICAO-TR] |

| | | | |
|---|---|---|---|
| **secure messaging - message authentication code** | **Retail MAC** | **112 bits** | **[ICAO-TR]** |

.

*Application Note:*

This SFR requires the TOE to implement the cryptographic primitive for secure messaging with message authentication code over transmitted data. The secure messing is implemented by FCS_COP.1.1/CM. The related session keys are agreed between the TOE and the terminal as part of either the PACE protocol according to the FCS_CKM.1/DH_PACE (PACE-KMAC). Note that in accordance with **[ICAO-TR]** the (two-key) Triple-DES could be used in Retail mode for secure messaging.

**FCS_CKM.4/DH_PACE Cryptographic key destruction**

**FCS_CKM.4.1/DH_PACE** The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **zeroisation** that meets the following: **none**.

**FCS_RND.1/PACE Quality metric for random numbers**

**FCS_RND.1.1/PACE** The TSF shall provide a mechanism to generate random numbers that meet **The average Shannon entropy per internal random bit exceeds 0.999**.

#### 7.1.10.2 Class FIA Identification and Authentication

**FIA_AFL.1/PACE Authentication failure handling**

**FIA_AFL.1.1/PACE** The TSF shall detect when **an administrator configurable positive integer within 1 to 255** unsuccessful authentication attempts occur related to **authentication attempts using the PACE password as shared password**.

**FIA_AFL.1.2/PACE** When the defined number of unsuccessful authentication attempts has been **met**, the TSF shall **wait for an increasing time between receiving of the terminal challenge and sending of the TSF response during the PACE authentication attempts**.

**FIA_UID.1/PACE Timing of identification**

**FIA_UID.1.1/PACE** The TSF shall allow

- **to establish the communication channel,**

- **carrying out the PACE Protocol according to [ICAO-TR],**
- **to read the Initialization Data if it is not disabled by TSF according to FMT_MTD.1/INI_DIS,** on behalf of the user to be performed before the user is identified.

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

*Application Note:*

In case of MRTD, the application should

- carry out the Chip Authentication Protocol v.1 according to **[ICAO-TR]**,
- carry out the Terminal Authentication Protocol v.1 according to **[ICAO-TR]**.

**FIA_UAU.1/PACE Timing of authentication**

**FIA_UAU.1.1/PACE** The TSF shall allow

- **to establish the communication channel,**
- **carrying out the PACE Protocol according to [ICAO-TR]**
- **to read the Initialization Data if it is not disable by TSF according to FMT_MTD.1/INI_DIS,**
- **to identify themselves by selection of the authentication key**
- **To carry out the authentication of the Manufacturer and Personalization Agent**

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

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

*Application Note:*

In case of MRTD, the application should

- carry out the Chip Authentication Protocol v.1 according to **[TR_03110]**
- carry out the Terminal Authentication Protocol according to **[TR_03110]**

**FIA_UAU.4/PACE Single-use authentication mechanisms**

**FIA_UAU.4.1/PACE** The TSF shall prevent reuse of authentication data related to

- **PACE Protocol according to [ICAO-TR],**
- **Authentication Mechanisms based on Triple-DES and AES**.

**FIA_UAU.5/PACE Multiple authentication mechanisms**

**FIA_UAU.5.1/PACE** The TSF shall provide

- **PACE Protocol according to [ICAO-TR],**
- **Secure messaging in MAC-ENC mode according to [ICAO-TR]**
- **Symmetric Authentication Mechanism based on Triple-DES and AES**

to support user authentication.

**FIA_UAU.5.2/PACE** The TSF shall authenticate any user's claimed identity according to the **following rules:**

- **Having successfully run the PACE protocol the TOE accepts only received commands with correct message authentication code sent by means of secure messaging with the key agreed with the terminal by means of the PACE protocol.**
- **The TOE accepts the authentication attempt as Personalisation Agent by the Authentication Mechanism with Personalization Agent Key(s).**
- **The TOE accepts only received commands with correct message authentication code sent by means of secure messaging with key agreed**.

*Application Note:*

When the secure messaging is automatically configured, the TOE automatically unwraps and wraps APDUs with FCS_COP.1/CM. For MRTD application: After run of the Chip Authentication Protocol Version 1 the TOE accepts only received commands with correct message authentication code sent by means of secure messaging with key agreed with the terminal by means of the Chip Authentication Mechanism v1.

**FIA_UAU.6/PACE Re-authenticating**

**FIA_UAU.6.1/PACE** The TSF shall re-authenticate the user under the conditions **each command sent to the TOE after successful run of the PACE protocol shall be verified as being sent by the PACE terminal**.

### 7.1.10.3 Class FTP Trusted Path/Channels

**FTP_ITC.1/PACE Inter-TSF trusted channel**

**FTP_ITC.1.1/PACE** 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/PACE** The TSF shall permit **another trusted IT product** to initiate communication via the trusted channel.

**FTP_ITC.1.3/PACE [Editorially Refined]** The TSF shall **enforce** communication via the trusted channel for **any data exchange between the TOE and the Terminal**.

*Application Note:*

To establish the trusted channel the TSF lies on Logical Channel according CoreG_LC security functional requirements. The Global Privacy Protocole (GPP) for PACE is dedicated to all Logical Channels, Local Privacy Protocole(LPP) for PACE is activated for applet communication on only one Logical Channel. The trusted channel can uses automatic secure messaging.

#### 7.1.10.4 Class FMT Security Management

**FMT_SMR.1/PACE Security roles**

**FMT_SMR.1.1/PACE** The TSF shall maintain the roles

- **Terminal,**
- **PACE authenticated BIS-PACE**
- **Manufacturer (Perso),**
- **Personalization Agent (Perso),**.

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

**FMT_SMF.1/PACE Specification of Management Functions**

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

- **Configuration.**
- **Initialization,**
- **Pre-personalization,**
- **Personalization**.

**FMT_LIM.1/PACE_Perso Limited capabilities**

**FMT_LIM.1.1/PACE_Perso** The TSF shall be designed in a manner that limits their capabilities so that in conjunction with 'Limited availability (FMT_LIM.2)' the following policy

is enforced **Deploying test features after TOE delivery do not allow 1.User Data to be manipulated and disclosed, 2.TSF data to be manipulated or disclosed, 3.software to be reconstructed, 4. substantial information about construction of TSF to be gathered which may enable other attacks**

*Application Note:*

The term "software" in item 4 refers to both IC Dedicated and IC Embedded Software.

**FMT_LIM.2/PACE_Perso Limited capabilities**

**FMT_LIM.2.1/PACE_Perso** The TSF shall be designed in a manner that limits their availability so that in conjunction with 'Limited capabilities (FMT_LIM.1)' the following policy is enforced **Deploying Test Features after TOE Delivery does not allow**

**1.User Data to be manipulated and disclosed,**

**2.TSF data to be manipulated or disclosed,**

**3.software to be reconstructed,**

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

*Application Note:*

The term "software" in item 4 refers to both IC Dedicated and IC Embedded Software.

**FMT_MTD.1/INI_ENA Management of TSF data**

**FMT_MTD.1.1/INI_ENA** The TSF shall restrict the ability to **write** the **Initialisation Data and the Pre-personalisation Data** to **the Manufacturer**.

**FMT_MTD.1/INI_DIS Management of TSF data**

**FMT_MTD.1.1/INI_DIS** The TSF shall restrict the ability to **read out** the **Initialisation Data and the Pre-personalisation Data** to **the Personalisation Agent**.

**FMT_MTD.1/KEY_READ Management of TSF data**

**FMT_MTD.1.1/KEY_READ** The TSF shall restrict the ability to **read** the

- **PACE passwords,**
- **Manufacturer Keys**
- **Pre-personalization Agent Keys,**
- **Personalisation Agent Keys,**

- **Chip Authentication Private Key**

to **none**.

#### 7.1.10.5 Class FDP User Data Protection

**FDP_RIP.1/PACE Subset residual information protection**

**FDP_RIP.1.1/PACE** 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:

- **Session Keys (immediately after closing related communication session),**
- **the ephemeral private key ephem-SK picc -PACE (by having generated a DH shared secret K)**.

#### 7.1.10.6 Class FPT Protection of Security Functions

**FPT_EMS.1/PACE TOE Emanation**

**FPT_EMS.1.1/PACE** The TOE shall not emit **electromagnetic and current emissions** in excess of **intelligible threshold** enabling access to **Personalization Agent Key(s) and Applicative keys and sensitive data**.

**FPT_EMS.1.2/PACE** The TSF shall ensure **any users** are unable to use the following interface **TOE external interfaces available according to form factor** to gain access to **Personalization Agent Key(s) and Applicative keys and sensitive data**.

*Application Note:*

When application is MTRD; Applicative keys are Chip Authentication Private Key and Active Authentication Key, and sensitive data are EF.DG3 and EF.DG4.

### *7.1.11 Additional Security Functional Requirements for PACE package - PACE-CAM - MODULE*

**FIA_UID.1/PACE_CAM Timing of identification**

**FIA_UID.1.1/PACE_CAM** The TSF shall allow **additionally to FIA_UID.1/PACE**

**1. carrying out the PACE CAM protocol according to [ICAO-9303] part 11** on behalf of the user to be performed before the user is identified.

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

**FIA_UAU.1/PACE_CAM Timing of authentication**

**FIA_UAU.1.1/PACE_CAM** The TSF shall allow **in addition to FIA_UAU.1/PACE**

**1. carrying out the PACE CAM Protocol according to [ICAO-9303] part 11** on behalf of the user to be performed before the user is authenticated.

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

**FIA_UAU.4/PACE_CAM Single-use authentication mechanisms**

**FIA_UAU.4.1/PACE_CAM** The TSF shall prevent reuse of authentication data related to

**1. PACE CAM Protocol according to [ICAO-9303] part 11 in addition to FIA_UAU.4/PACE**.

**FIA_UAU.5/PACE_CAM Multiple authentication mechanisms**

**FIA_UAU.5.1/PACE_CAM** The TSF shall provide

**1. PACE CAM Protocol according to [ICAO-9303] part 11** to support user authentication.

**FIA_UAU.5.2/PACE_CAM** The TSF shall authenticate any user's claimed identity according to the **following rules:**

- **The same rules from FIA_UAU.5.2/PACE applies with the PACE CAM protocol and**
- **The TOE accepts the authentication attempt by means of the Terminal Authentication only if the terminal uses the public key presented during the Chip Authentication or the terminal uses the public key presented during PACE-CAM and the secure messaging established during PACE**.

**FIA_UAU.6/PACE_CAM Re-authenticating**

**FIA_UAU.6.1/PACE_CAM** The TSF shall re-authenticate the user under the conditions **each command sent to the TOE after successful run of the PACE CAM protocol shall be verified as being sent by the PACE Terminal**.

**FMT_MTD.1/PACE_CAM_KEY_READ Management of TSF data**

**FMT_MTD.1.1/PACE_CAM_KEY_READ** The TSF shall restrict the ability to **read** the

**a. PACE passwords**

**b. Modular invert of the CA key** to

**none**.

**FMT_MTD.1/PACE_CAM_KEY_WRITE Management of TSF data**

**FMT_MTD.1.1/PACE_CAM_KEY_WRITE** The TSF shall restrict the ability to **write** the **Modular invert of the CA key** to **Personalization Agent**.

### *7.1.12 Additional Security Functionnal Requirements for API - RSA - MODULE*

**FCS_CKM.1/RSA Cryptographic key generation**

**FCS_CKM.1.1/RSA** 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:**

| **Cryptographic key generation algorithm** | **Cryptographic key size** | **List of standards** |
|---|---|---|
| **RSA** | **from 1024 to 4096 bits with a step of 256-bits** | **ANSI X9.31** |

.

*Application Note:*

- The keys can be generated and diversified in accordance with **[JCAPI]** specification in classes KeyPair (at least Session key generation) and RandomData.
- This component shall be instantiated according to the version of the Java Card API applying to the security target and the implemented algorithms **[JCAPI]**.
- This component shall be instantiated according to the version of the Global Platform GP 2.3 **[GP1]**.

**FCS_COP.1/RSA Cryptographic operation**

**FCS_COP.1.1/RSA** 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 table below:**

| Cryptographic operation | Cryptographic algorithm | TOE supported size | "Standardized" size | List of standards |
|---|---|---|---|---|
| **signature, signature's verification, encryption and decryption** | **RSA CRT, private keys. RSA SFM (Straightforward) public and private keys.** | **from 2048 to 4096 bits with a step of 256-bits** | **2048 bits** | **ANSI X9.31, ISO/IEC 9796-1, annex A, section A.4 and A.5, and annex C, PKCS#1** |

.

*Refinement:*

RSA without CRT /IDEMIA has developed the algorithm using HW PK accelerator/Data Encryption and Decryption/RSA Without CRT Data /Between 1024 bits to 2048 bits/PKCS#1 V2.0; 1st October, 1998

RSA with CRT /IDEMIA has developed the algorithm using HW PK accelerator/Data Encryption and Decryption/RSA With CRT Data /Between 1024 bits and 2048 bits/PKCS#1 V2.0; 1st October, 1998 RNG/IDEMIA has developed the algorithm using HW RNG as seed/Random generator//No cryptographic key/FIPS SP800-90, 2007, March

*Application Note:*

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

### *7.1.13 Additional Security Functionnal Requirements for API - SHA3 - MODULE*

**FCS_COP.1/SHA3 Cryptographic operation**

**FCS_COP.1.1/SHA3** 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 table below:**

| Cryptographic operation | Cryptographic algorithm | TOE supported size | "Standardized" size | List of standards |
|---|---|---|---|---|
| **Hash functions** | **SHA-384** | **SHA-384** | **SHA-384** | **Secure Hash Standard, FIPS PUB 180-3** |

.

*Refinement:*

SHA /IDEMIA has developed the algorithm/Hash function/SHA-384/No cryptographic key/Secure Hash Standard, Federal Information Processing Standards Publication 180-3, 2008, october

*Application Note:*

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

### *7.1.14 Additional Security Functional Requirements for API - Biometrics (MOC) - MODULE*

**FIA_AFL.1/BIO Authentication failure handling**

**FIA_AFL.1.1/BIO** The TSF shall detect when **an administrator configurable positive integer within user defined maximum from 1 to 127 for D.BIO** unsuccessful authentication attempts occur related to **any user authentication using MOC**.

**FIA_AFL.1.2/BIO** When the defined number of unsuccessful authentication attempts has been **met**, the TSF shall **block the MOC**.

**FMT_MTD.1/BIO Management of TSF data**

**FMT_MTD.1.1/BIO** The TSF shall restrict the ability to **change_default, modify and query** the **D.BIO** to **applet itself**.

*Application Note:*

The D.BIO is usable for matching only at the end of biometric initialization process.

### *7.1.15 Additional Security Functional Requirements for PACE package - PACE-DH - MODULE*

**FCS_CKM.1/DH_PACE_DH Cryptographic key generation**

**FCS_CKM.1.1/DH_PACE_DH** The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **Diffie-Hellman-Protocol** and specified cryptographic key sizes **2048-4096 bits in steps of one bit** that meet the following: **PKCS#3 and [TR_03110]**.

*Application Note:*

The TOE generates a shared secret value with the terminal (For a MRTD application: during the Chip Authentication Protocol Version 1, see [TR-03110]). This protocol is based on the Diffie-Hellman-Protocol compliant to PKCS#3, The shared secret value K is used for deriving the AES or DES session keys for message encryption and message authentication according to [48] for the TSF required by FCS_COP.1/PACE_ENC and FCS_COP.1/PACE_MAC. FCS_CKM.1/DH_PACE_DH implicitly contains the requirements for the hashing functions used for key derivation by demanding compliance to **[ICAO-TR]**.

### *7.1.16 Additional Security Functionnal Requirements for HMAC - Module*

**FCS_COP.1/HMAC Cryptographic operation**

**FCS_COP.1.1/HMAC** 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 table below:**

| Cryptographic operation | Cryptographic algorithm | TOE supported size | "Standardized" size | List of standards |
|---|---|---|---|---|
| signature | HMAC | 64 bits up to 1016 bits Based on SHA-256, SHA-384 and SHA-512 | SHA-2 with key size > 128 bits | FIPS 198 The Keyed-Hash Message Authentication Code (HMAC) |

## 7.2 Security Assurance Requirements

The Evaluation Assurance Level is EAL6 augmented with ALC_FLR.1.

The Security policy modelling (ADV_SPM) family defines the requirements for formally modelling selected SFRs, and providing correspondence between the functional specification and the formal model.

ADV_SPM.1.1D: The developer shall provide a formal security policy model for the [FIREWALL access control SFP].

## 7.3 Security Requirements Rationale

### *7.3.1 Objectives*

#### 7.3.1.1 Security Objectives for the TOE

**IDENTIFICATION**

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

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

**EXECUTION**

**O.FIREWALL** This objective is met by the FIREWALL access control policy FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL, the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM), the 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, S, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM,) also indirectly contribute to meet this objective.

This objective is also covered by the following additional SFRs:

- Stack control (*/RV_Stack): FDP_ACC.2/RV_Stack, FDP_ACF.1/RV_Stack, FMT_MSA.1/RV_Stack, FMT_MSA.2/RV_Stack, FMT_MSA.3/RV_Stack, FMT_SMF.1/RV_Stack
- Heap control (*/RV_Heap): FDP_ACC.2/RV_Heap, FDP_ACF.1/RV_Heap, FMT_MSA.1/RV_Heap, FMT_MSA.2/RV_Heap, FMT_MSA.3/RV_Heap, FMT_SMF.1/RV_Heap
- Transient control (*/RV_Transient): FDP_ACC.2/RV_Transient, FDP_ACF.1/RV_Transient, FMT_MSA.1/RV_Transient, FMT_MSA.2/RV_Transient, FMT_MSA.3/RV_Transient, FMT_SMF.1/RV_Transient

For each of those control, the SFR define the access control (FDP_ACC and FDP_ACF), the operation (FMT_MSA) and the role (FMT_SMF).

The Stack control enforces O.FIREWALL by defining additional rules, such as the control of the stack is more precise. Information is provided in the application note.

The Heap control enforces O.FIREWALL by defining additional rules, such as the heap usage is improved. Information is provided in the application note.

The Transient enforces O.FIREWALL by defining additional rules, such as the heap usage is improved. Information is provided in the application note.

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

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

**O.NATIVE** O.NATIVE This security objective is covered by FDP_ACF.1/FIREWALL: the only means to execute native code is the invocation of a Java Card API method or by the execution of the TPL application under the access control of the 2 sfrs: FDP_ACC.2/JBox and FDP_ACF.1/JBox.

**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).

The FPT_RCV.4/SCP contributes to the objective O.OPERATE as it ensures that when reading or writing operations are interrupted by power loss, the operation either is completed or recovers in a consistent and secure state.

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

*Application note:* Startup of the TOE (TSF-testing) can be covered by FPT_TST.1. This SFR component is not mandatory in **[JCRE]**, but appears in most of security requirements

documents for masked applications. Testing could also occur randomly. Self-tests may become mandatory in order to comply with FIPS certification [FIPS 140-2].

**O.REALLOCATION** This security objective is satisfied by the following SFRs: FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray, 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 SFRs 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 SFRs (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).

**O.ARRAY_VIEWS_CONFID** Array views have security attributes of temporary objects where the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) prevents an application from storing a reference to the array view. Furthermore, array views may not have ATTR_READABLE_VIEW security attribute which ensures that no application can read the contents of the array view.

**O.ARRAY_VIEWS_INTEG** Array views have security attributes of temporary objects where the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) prevents an application from storing a reference to the array view. Furthermore, array views may not have ATTR_WRITABLE_VIEW security attribute which ensures that no application can alter the contents of the array view.

**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/RSA, FCS_COP.1/RSA (when RSA module is embedded), FCS_COP.1/SHA3 (when SHA3 module is embedded), FCS_COP.1/HMAC (when HMAC module is embedded), FCS_CKM.1, FCS_CKM.4, and FCS_COP.1, FCS_COP.1/PP, FCS_COP.1/CM. FPR_UNO.1 and FPR_UNO.1/USE_KEY contributes in covering this security objective and controls the observation of the cryptographic operations which may be used to disclose the keys. When PACE package is

used, O.CIPHER supports FCS_COP.1/PACE_ENC, FCS_COP.1/PACE_MAC and FCS_COP.1/PACE_ to ensure the secure messaging implemented by FCS_COP.1/CM.

**O.RNG** This security objective is directly covered by FCS_RNG.1 which ensures the cryptographic quality of random number generation.

**O.KEY-MNGT** This relies on the same security functional requirements as O.CIPHER, plus FDP_RIP.1, FPT_TDC.1/CM and FDP_SDI.2/DATA as well. Precisely it is met by the following components: FCS_CKM.1/RSA, FCS_COP.1/RSA (when RSA module is embedded), FCS_COP.1/SHA3 (when SHA3 module is embedded), FCS_COP.1/HMAC (when HMAC module is embedded), FCS_CKM.1, FCS_CKM.4, FCS_COP.1, FCS_COP.1/PP, FPR_UNO.1, FPR_UNO.1/USE_KEY, FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL and FDP_RIP.1/TRANSIENT.

**O.PIN-MNGT** This security objective is ensured by FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray, 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/DATA 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. FIA_AFL1.1/CM, FIA_AFL.1/PIN and FIA_AFL.1/GP_PIN ensure the objective regarding authentications failures. FMT_MTD.1/PIN and FMT_MTD.2/GP_PIN ensure the objective regarding the management of the TSF data.

**O.TRANSACTION** Directly met by FDP_ROL.1/FIREWALL, FDP_RIP.1/ABORT, FDP_RIP.1/ODEL, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray, 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 CAP FILE 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).

**Additional security objectives for the TOE**

**O.SCP.SUPPORT** The components FPT_RCV.4/SCP (SCP stands for smart card platform) are used to support the objective O.SCP.SUPPORT to assist the TOE to recover in the event of a power failure. If the power fails or the card is withdrawn prematurely from the CAD the operation of the TOE may be interrupted leaving the TOE in an inconsistent state.

All the Crypto SFRS support this objective as they provide secure low-level cryptographic processing to the Java Card System and Global Platform:

- FCS_CKM.1/RSA, FCS_COP.1/RSA (when RSA module is embedded),
- FCS_COP.1/SHA3 (when SHA3 module is embedded),
- FCS_COP.1/HMAC (when HMAC module is embedded),
- FCS_CKM.1, FCS_CKM.4, FCS_COP.1,
- FCS_COP.1/CM,
- FCS_COP.1/PP, FCS_CKM.4/PP

All the FSRs related to the Firewall contribute to the realization of the objective.

The FDP_ROL.1 Firewall ensures the rollback of some operations within the specified scope as defined in the ROL.1.2/Firewall.

When PACE package is used, this objective supports FCS_COP.1/PACE_ENC and FCS_COP.1/PACE_MAC to ensure the secure messaging implemented by FCS_COP.1/CM.

*Application Note:* all SFRs related to O.OPERATE and O.ALARM support the O.SCP.SUPPORT

**O.SCP.IC** This objective is met by the component FPT_PHP.3/SCP for the IC resistance and FCS_RNG.1 for RNG quality.

**O.SCP.RECOVERY** The component FPT_RCV.4/SCP is used to support the objective O.SCP.RECOVERY to assist the TOE to recover in the event of a power failure. If the power fails or the card is withdrawn prematurely from the CAD the operation of the TOE may be

interrupted leaving the TOE in an inconsistent state. This objective is met by the components FPT_FLS.1, FAU_ARP.1 and FRU_FLT.1/SCP.

**O.RESIDENT_APPLICATION** This objective is covered by the following set of SFR:

- Access control: FDP_ACC.2/PP, FDP_ACF.1/PP, FDP_UCT.1/PP and FDP_ITC.1/PP
- Rules for authentication: FIA_AFL.1/PP, FIA_UAU.1/PP, FIA_UAU.1/CM
- Security Management: FMT_MSA.1/PP, FMT_SMF.1/PP, FMT_MOF.1/PP, FMT_SMR.2/PP, FMT_MSA.3/PP and FMT_SMR.2/CM
- Once the ISK is loaded, the RA ensures MSK and LSK are no more available thanks to Cryptographic Key Destruction: FCS_CKM.4/PP.

This objective is also covered by Card Manufacturer authentication: Rules for authentication: FIA_UAU.7/CardIssuer, FIA_UAU.4/CardIssuer, FIA_UAU.4/CardManu, FIA_UAU.7/CardManu.

This objective is also covered by FLEXICODE with rules for deleting the modules: FDP_ACC.2/FlxC, FDP_ACF.1/FlxC, FMT_MOF.1/FlxC, FPT_FLS.1/FlxC, FDP_RIP.1/FlxC

**O.CARD_MANAGEMENT** This objective is fulfilled by the following set of SFR:

The FDP_ACC.2/ADEL and FDP_ACF.1/ADEL contribute to meet the ADEL access control policy that ensures the non-introduction of security holes.

The FDP_RIP.1/ADEL ensure that the deleted information is not accessible.

The FMT_MSA.1/ADEL ensures restrict the ability to modify the secure attributes the FMT_MSA.3/ADEL ensures the assignment of restrictive values.

The FMT_SMR.1/ADEL and FMT_SMR.2/PP maintains the role of the applet deletion manager.

The FPT_FLS.1/ADEL contributes to the objective by protecting the TSFs against possible failures of the deletion procedure.

The 2 SFRs FPT_RCV.3/Installer and FPT_FLS.1/Installer contributes to meet the objective by protecting the TSFs from failures of the deletion procedure.

The SFR FDP_UIT.1/CM contributes by enforcing the Secure Channel Protocol Information flow control policy and the Security Domain access control policy which control the integrity of the corresponding data.

The SFR FIA_UID.1/CM testes if the Secure Channel is open to allow card management operations.

The SFR FDP_IFF.1/CM ensures the access control policy for the loaded data (as packages).

The FCO_NRO.2/CM this SFR ensures the origin of the load file. It verifies the identity of the origin of the load file before start the loading.

FCO_NRO.2/CM_DAP this SFR generates an evidence of the origin of the transmitted load file during CAP File loading.

The FDP_IFC.2/CM, this SFR ensures that loading commands are issued in the Secure Channel session.

The SFR FDP_ROL.1/FIREWALLensures that the card management operations are cleaned aborted.

The SFR FDP_ITC.2/Installer enforces the Firewall access control policy and flow control policy when importing card management data.

The SFR FPT_FLS.1/ODEL ensures the preservation of secure state when failures occur.

The SFR FMT_MSA.1/CM ensures the management of the security attributes to the card manager, for the modification of the life cycle of the card, the keyset version and value,…

The SFR FMT_MSA.3/CM, this SFR ensures that the security attributes can only be changed by the card manager.

The SFR FMT_SMF.1/CM and FMT_SMF.1/PP Only the card manger is able to modify the security attributes of the management functions. The security role is specified in the FMT_SMR.1/CM and FMT_SMR.2/CM.

The SFR FPT_TDC.1/CM ensure that key sets and packages loaded are well under key management.

The SFR FTP_ITC.1/CM ensures the trusted Channel Communications.

FIA_AFL.1/CM, FIA_UAU.1/CM, FIA_UAU.4/CardIssuer, and FIA_UAU.7/CardIssuer, ensure the authentication of the card issuer before gaining access to management operations.

FIA_UAU.4/CardManu and FIA_UAU.7/CardManu ensure the authentication of the card manufacturer before gaining access to management operations. The FPR_UNO.1/Key_CM ensures the un-observability of the CM key when imported..

The FPT_TST.1 This TSF contributes to ensure the correct operation of the card management functions as it tests the integrity of the TSF functions during initial start-up.

The SFR FPT_TDC.1/CM ensures that key sets and packages loaded are well under key management.

**O.SECURE_COMPARE** This objective is fulfilled by FDP_SDI.2/DATA. It ensures that comparison is confidential.

**O.PATCH_LOADING Authentication of the entity loading the patch by the TOE**

FDP_ACC.2/PP, FDP_ACF.1/PP, FIA_UAU.1/PP and FIA_UID.1/PP provide access control for patch loading. The subject entitled to load the patch – the card manufacturer - is authenticated by the TOE thanks to FCS_COP.1/PP. Wrong authentication of the Card manufacturer agent are detected thanks to FIA_AFL.1/PP

**Authentication of the TOE**

To avoid impersonation of the TOE by a fake chip, the TOE authenticates itself; from phase 6 (after patch loading) with FTP_ITC.1/CM and FCS_COP.1/CM thanks to the TOE authentication key (ISK/KMC). From phase 6, the TOE authentication is required prior to any trusted channel establishment with FTP_ITC.1/CM (data sent by the TOE must be decrypted to carry on the authentication).

The TOE authentication key (ISK/KMC) is securely loaded in phase 4/5 protected in confidentiality with FDP_UCT.1/PP and integrity with FDP_UIT.1/PP through the trusted channel established by the Card Manufacturer with FDP_ITC.1/PP. The trusted channel and the TOE authentication key (ISK/KMC) encryption is supported by FCS_COP.1/PP that relies on the TOE's MSK which is the first key present in the TOE.

**Diversification of keys**

The TOE's MSK used to authenticate the Card manufacturer is derived from the MSK before the first use. The MSK is loaded in the TOE in phase 1 (covered by [ALC]).

**Integrity, confidentiality and authenticity of the patch during loading**

Patch loading is performed in a confidential manner with FDP_UCT.1/PP and protected in integrity and confidentiality with FDP_UIT.1/PP. Confidentiality, integrity and authenticity of the patch loading is supported by cryptographic mechanisms supported by FCS_COP.1/PP.

Patch data to be written in the TOE have been prior encrypted by the TOE developer using LSK key. Once these data loaded, the integrity (SHA256) of the modified code is updated and compared to the provided one in the patch package.

**Irreversible locking of the patch loading features**

The patch can be loaded in phase 4 and 5 of the TOE's life cycle. At the end of phase 5, FMT_MOF.1/PP ensures the possible irreversible locking once in OP_READY, after pre-production state. Once in OP_READY state, those APDU cannot be used if the lock has been configured. In the other case, the patches can be loaded at user phase.

**Erasure of the key used**

FCS_CKM.4/PP ensures the secure destruction of the keys involved in the patch loading mechanism (LSK and MSK) at the end of phase 5.

**Identification of the patch after loading**

Once loaded and during the rest of the TOE life cycle, the identification and authentication (unique identifier of the patch) of the patch, being a part of the TOE is provided by FAU_STG.2. When requested, the identification and authentication data (of entire code, including patch) are dynamically retrieved from the patch code stored in the non-volatile memory of the TOE.

**Integrity check before usage of the patch**

At start up, the integrity of the entire code, patch included, is checked by the TOE through self-tests provided by FPT_TST.1. In case the computed signature differs from the one stored in NVM, an integrity error is detected and a killcard is raised.

**O.JBox_FW** The access control mechanisms described by O.JBox_FW are addressed by the SFP defined by the security functional requirements FDP_ACC.2/JBox, FDP_ACF.1/JBox, FMT_MSA.1/JBox; FMT_MSA.3/JBox and FMT_SMF.1/JBox.

**O.FLEXICODE** The access control mechanisms described by O.FLEXICODE are addressed by the SFP defined by the security functional requirements FDP_ACC.2/FlxC, FDP_ACF.1/FlxC, FMT_MSA.1/FlxC, FMT_MSA.2/FlxC, FMT_MSA.3/FlxC, FMT_MSA.4/FlxC, FMT_MOF.1/FlxC and FMT_SMF.1/FlxC. The preservation of a genuine source code is ensures by FPT_FLS.1/FlxC and FDP_RIP.1/FlxC.

**Integrity check before usage of the patch**

At start up, the integrity of the entire code, patch included, is checked by the TOE through self-tests provided by FPT_TST.1. In case the computed signature differs from the one stored in NVM, an integrity error is detected and a killcard is raised.

**O.DUMP_PERSO Personalization dump image confidentiality and integrity**

**Access to personalization data** FMT_SMR.2/CM ensures R.personaliser authentication and OP_READY state of card life cycle for extracting memory blocks.

FDP_ACC.2/FIREWALL, FDP_ACC.2/PP, FDP_ACF.1/FIREWALL, FIA_UAU.1/PP and FIA_UID.1/PP provide access control personalization memory blocks. The subject entitled to load the image – the card manufacturer - is authenticated by the TOE thanks to

FCS_COP.1/PP. Wrong authentication of the Card manufacturer agent are detected thanks to FIA_AFL.1/CM

**Compute confidentiality** Confidentiality, integrity and authenticity of the image extracting (encipher) and loading (decipher)is supported by cryptographic mechanisms supported by FCS_COP.1/PP. Image loading is performed in a confidential manner with FDP_UCT.1/PP and protected in integrity and confidentiality with FDP_SDI.2/DATA and FDP_UIT.1/PP.

**Integrity check before usage of the perso dump** Memory Blocks data to be written in the TOE have been prior encrypted by the TOE using DSK key. Once these data loaded, the integrity (checksum) of the image is compared to the provided one in the image data.

**Authentication of the TOE** To avoid impersonation of the TOE by a fake chip, the TOE authenticates itself; from phase 6 (after image loading) with FTP_ITC.1/CM and FCS_COP.1/CM thanks to the TOE authentication key (ISK/KMC). From phase 6, the TOE authentication is required prior to any trusted channel establishment with FTP_ITC.1/CM (data sent by the TOE must be decrypted to carry on the authentication).

The TOE authentication key (ISK/KMC) is securely loaded in phase 4/5 protected in confidentiality with FDP_UCT.1/PP and integrity with FDP_UIT.1/PP through the trusted channel established by the Card Manufacturer with FDP_ITC.1/PP. The trusted channel and the TOE authentication key (ISK/KMC) encryption is supported by FCS_COP.1/PP that relies on the TOE's MSK which is the first key present in the TOE.

**Integrity check before usage of the peronaliaztion data** At start up, the integrity of the entire code, personalization data included, is checked by the TOE through self-tests provided by FPT_TST.1. In case the computed signature differs from the one stored in NVM, an integrity error is detected and a killcard is raised.

**O.CLFDB_DECIPHER** This security objective is directly covered by FCS_COP.1/CM. FPR_UNO.1 and FPR_UNO.1/USE_KEY contributes in covering this security objective and controls the observation of the cryptographic operations which may be used to disclose the keys. FCS_CKM.4 contributes to the key destruction after all blocks imported. In the scope of National Identity Card, for instance, a need to install ciphered applet after wafer delivery implied to encrypt the blocks for applet installation: the FDP_ITC.2/Installer contributes to the objective.

**Security Objectives for the TOE with Monotonic counters package**

**O.MTC-CTR-MNGT** O.MTC-CTR-MNGT This security objective is ensured by FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FDP_ROL.1/FIREWALL and FDP_SDI.2/MONOTONIC_COUNTER 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. Note that the objective applies only to configurations including the javacardx.security.util package defined in **[JCAPI]**.

**Security Objectives for the TOE with PACE package**

**O.PACE_Data_Integrity** The security objective O.PACE_Data_Integrity "Application data" requires the TOE to protect the integrity of the application data requiring usage of PACE

(e.g. logical electronic document) stored on the TOE against physical manipulation and unauthorized writing. Physical manipulation is addressed by FPT_PHP.3/SCP.The Personalisation Agent must identify and authenticate themselves according to FIA_UID.1/PACE and FIA_UAU.1/PACE before accessing these data. FIA_UAU.4/PACE, FIA_UAU.5/PACE and FCS_CKM.4/DH_PACE represent some required specific properties of the protocols used. The SFR FMT_SMR.1/PACE manages the roles and the SFR FMT_SMF.1/PACE manages the TSF management functions. Unauthorised modifying of the exchanged data is addressed, in the first line, by FTP_ITC.1/PACE using FCS_COP.1/PACE_MAC. For PACE secured data exchange, a prerequisite for establishing this trusted channel is a successful PACE Authentication (FIA_UID.1/PACE, FIA_UAU.1/PACE) using FCS_CKM.1/DH_PACE or FCS_CKM.1/DH_PACE_DH and possessing the special properties FIA_UAU.5/PACE, FIA_UAU.6/PACE. FIA_AFL.1/PACE allows to manage errors in PACE secure channel management. FDP_RIP.1/PACE requires erasing the values of session keys (here: for KMAC). The session keys are destroyed according to FCS_CKM.4/DH_PACE after use. The SFR FCS_RND.1/PACE represents a general support for cryptographic operations needed. In pre-personalisation, the SFR FCS_CKM.1/DH_PACE and FCS_COP.1/PACE_MAC ensure the integrity of data transfers after successful authentication of the pre-personalisation agent according to FIA_UID.1/PACE and FIA_UAU.1/PACE, with the support of FIA_AFL.1/PACE. The SFR FMT_MTD.1/KEY_READ requires that data cannot be unauthorized read afterwards.

Justification for PACE-CAM: The Personalisation Agent must identify and authenticate themselves according to FIA_UID.1/PACE_CAM and FIA_UAU.1/PACE_CAM for PACE CAM protocol before accessing these data. FIA_UAU.4/PACE_CAM, FIA_UAU.5/PACE_CAM for PACE CAM protocol and FCS_CKM.4 represent some required specific properties of the protocols used. Secured data exchange, a prerequisite for establishing this trusted channel is a successful PACE Authentication FIA_UID.1/PACE_CAM, FIA_UAU.1/PACE_CAM PACE_CAM Authentication using FCS_CKM.1/DH_PACE or FCS_CKM.1/DH_PACE_DH and possessing the special properties FIA_UAU.5/PACE_CAM, FIA_UAU.6/PACE_CAM for PACE CAM. The SFR FMT_MTD.1/PACE_CAM_KEY_WRITE and FMT_MTD.1/PACE_CAM_KEY_READ requires that the modular invert of the CA key cannot be written unauthorized or read afterwards.

**O.PACE_Data_Authenticity** The security objective O.PACE_Data_Authenticity aims ensuring authenticity of the User- and TSF data (after the PACE Authentication) by enabling its verification at the terminal-side and by an active verification by the TOE itself.This objective is mainly achieved by FTP_ITC.1/PACE using FCS_COP.1/PACE_MAC. A prerequisite for establishing this trusted channel is a successful PACE (FIA_UID.1/PACE, FIA_UAU.1/PACE) using FCS_CKM.1/DH_PACE or FCS_CKM.1/DH_PACE_DH resp. FDP_RIP.1/PACE requires erasing the values of session keys (here: for KMAC). FIA_UAU.4/PACE, FIA_UAU.5/PACE, FIA_UAU.6/PACE and FCS_CKM.4/DH_PACE represent some required specific properties of the protocols used. The SFR FMT_MTD.1/KEY_READ restricts the access to the PACE passwords and the Chip Authentication Private Key. The SFR FCS_RND.1/PACE represents a general support for cryptographic operations needed.The SFRs FMT_SMF.1/PACE and FMT_SMR.1/PACE support the functions and roles related. FIA_AFL.1/PACE allows to manage errors in PACE secure channel management.

Justification for PACE-CAM: A prerequisite for establishing this trusted channel is a successful PACE-CAM authentication FIA_UID.1/PACE_CAM, FIA_UAU.1/PACE_CAM using FCS_CKM.1/DH_PACE or FCS_CKM.1/DH_PACE_DH and possessing the special properties FIA_UAU.5/PACE_CAM, FIA_UAU.6/PACE_CAM. FIA_UAU.4/PACE_CAM,

FIA_UAU.5/PACE_CAM and FCS_CKM.4 represent some required specific properties of the protocols used. FMT_MTD.1/PACE_CAM_KEY_READ restricts the access to the PACE passwords and the modular invert of the Chip Authentication key.

**O.PACE_Data_Confidentiality** The security objective O.PACE_Data_Confidentiality aims that the TOE always ensures confidentiality of the User- and TSF-data stored and, after the PACE Authentication of these data exchanged.FIA_UAU.4/PACE, FIA_UAU.5/PACE, FIA_UAU.6/PACE and FCS_CKM.4/DH_PACE represent some required specific properties of the protocols used. This objective for the data exchanged is mainly achieved by FTP_ITC.1/PACE using FCS_COP.1/PACE_ENC. A prerequisite for establishing this trusted channel is a successful PACE (FIA_UID.1/PACE, FIA_UAU.1/PACE) using FCS_CKM.1/DH_PACE or FCS_CKM.1/DH_PACE_DH. FDP_RIP.1/PACE requires erasing the values of session keys (here: for Kenc). FIA_AFL.1/PACE allows to manage errors in PACE secure channel management. The SFR FMT_MTD.1/KEY_READ restricts the access to the PACE passwords and the Chip Authentication Private Key. The SFR FCS_RND.1/PACE represents the general support for cryptographic operations needed.The SFRs FMT_SMF.1/PACE and FMT_SMR.1/PACE support the functions and roles related.In Perso/pre-personalisation, the SFR FCS_CKM.1/DH_PACE,FCS_CKM.1/DH_PACE_DH, FCS_COP.1/PACE_MAC and FCS_COP.1/PACE_ENC ensure the confidentiality of data transfers after successful authentication of the pre-personalisation agent according to FIA_UID.1/PACE and FIA_UAU.1/PACE, with the support of FIA_AFL.1/PACE.

Justification for PACE-CAM: This objective for the data stored is mainly achieved by FIA_UAU.4/PACE_CAM, FIA_UAU.5/PACE_CAM and FCS_CKM.4 represent some required specific properties of the protocols used. A prerequisite for establishing this trusted channel is a successful PACE-CAM authentication with FIA_UID.1/PACE_CAM, FIA_UAU.1/PACE_CAM using FCS_CKM.1/DH_PACE and possessing the special properties FIA_UAU.5/PACE_CAM, FIA_UAU.6/PACE_CAM for PACE CAM. FMT_MTD.1/PACE_CAM_KEY_READ restricts the access to the PACE passwords and the modular invert of the Chip Authentication key.

**O.PACE_Prot_Abuse-Func** The security objective O.PACE_Prot_Abuse_Func "Protection against Abuse of Functionality" is ensured by the SFR FMT_LIM.1/PACE_Perso_Perso and FMT_LIM.2/PACE_Perso_Perso which prevent misuse of test functionality of the TOE or other features which may not be used after TOE Delivery.

**O.PACE_Prot_Inf_Leak** The security objective O.PACE_Prot_Inf_Leak "Protection against Information Leakage" requires the TOE to protect confidential TSF data stored and/or processed in the electronic document's chip against disclosure

- by measurement and analysis of the shape and amplitude of signals or the time between events found by measuring signals on the electromagnetic field, power consumption, clock, or I/O lines which is addressed by the SFR FPT_EMS.1/PACE,
- by forcing a malfunction of the TOE which is addressed by the SFR FPT_FLS.1 and FPT_TST.1, and/or
- by a physical manipulation of the TOE which is addressed by the SFR FPT_PHP.3/SCP.

**O.PACE_Prot_Phys-Tamper** The security objective O.PACE_Prot_Phys-Tamper "Protection against Physical Tampering" is covered by the SFR FPT_PHP.3/SCP.

**O.PACE_Prot_Malfunction** The security objective O.PACE_Prot_Malfunction "Protection against Malfunctions" is covered by (i) the SFR FPT_TST.1 which requires self tests to demonstrate the correct operation and tests of authorized users to verify the integrity of TSF data and TSF code, and (ii) the SFR FPT_FLS.1 which requires a secure state in case of detected failure or operating conditions possibly causing a malfunction.

**O.PACE_Identification** The security objective O.PACE_Identification "Identification of the TOE" addresses the storage of Initialisation and Pre-Personalisation Data in its non-volatile memory, whereby they also include the IC Identification Data uniquely identifying the TOE's chip. This will be ensured by TSF according to SFR.1. The SFR FMT_MTD.1/INI_ENA allows only the Manufacturer to write Initialisation and Pre-personalisation Data (including the Personalisation Agent key). The SFR FMT_MTD.1/INI_DIS requires the Personalisation Agent to disable access to Initialisation and Pre-personalisation Data in the life cycle phase 'operational use'. The SFRs FMT_SMF.1/PACE and FMT_SMR.1/PACE support the functions and roles related.

**O.PACE_AC_Pers** The security objective O.PACE_AC_Pers "Access Control for Personalization" The TOE must ensure that the TOE and Applicative data (e.g.PACE data and MRTD data (if any) e.g. logical travel document data in EF.DG1 to EF.DG16, the Document Security Object according to LDS [ICAO_9303] and the TSF data can be written by authorized Personalisation Agents only. The TOE and Applicative data (e.g. logical travel document data in EF.DG1 to EF.DG16) and the TSF data may be written only during and cannot be changed after personalisation phase. The SFR FMT_SMR.1/PACE manages the roles (including Personalization Agent) and the SFR FMT_SMF.1/PACE lists the TSF management functions (including Personalization). The SFR FMT_MTD.1/INI_ENA allows only the Manufacturer to write Initialisation and Pre-personalisation Data (including the Personalisation Agent key). The SFR FMT_MTD.1/INI_DIS requires the Personalisation Agent to disable access to Initialisation and Pre-personalisation Data in the life cycle phase 'operational use'. The SFR FMT_MTD.1/KEY_READ and FMT_MTD.1/PACE_CAM_KEY_WRITE requires that data cannot be unauthorized read or write afterwards. O.PACE_AC_Pers Access Control for Personalisation of TOE and Applicative data

Justification for PACE_CAM: The write access to the logical electronic document data are defined by the SFR FIA_UID.1/PACE_CAM for PACE CAM,FIA_UAU.1/PACE_CAM, FDP_ACC.1/TRM and FDP_ACF.1/TRM in the same way: only the successfully authenticated Personalisation Agent is allowed to write the data of the groups EF.DG1 to EF.DG16 of the logical electronic document only once. The SFRs FMT_MTD.1/PACE_CAM_KEY_READ and FPT_EMS.1 restrict the access to the Personalisation Agent Keys, the Chip Authentication Private Key and the Active Authentication Private key. The authentication of the terminal as Personalisation Agent shall be performed by TSF according to SFRs FIA_UAU.4/PACE_CAM and FIA_UAU.5/PACE_CAM.

**Additional security objectives for the TOE with Sensitive Array package**

**O.SENSITIVE_ARRAYS_INTEG** O.SENSITIVE_ARRAYS_INTEG This security objective is covered directly by the SFR FDP_SDI.2/ARRAY Integrity_Senstive_Array which ensures that integrity errors related to the user data stored in sensitive arrays are detected by the TOE.

**Additional security objectives for the TOE with Biometrics (MOC) - MODULE**

**O.BIO-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/DATA 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. FIA_AFL.1/CM and FIA_AFL.1/BIO ensure the objective regarding authentications failures. FMT_MTD.1/BIO ensures the objective regarding the management of the TSF data.

**Additional security objectives for the TOE with DBI - MODULE**

**O.DBI-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/DATA 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. FIA_AFL.1/CM ensure the objective regarding authentications failures linked to applets. FMT_MTD.1/Activate_DBI and FMT_MTD.1/Deactivate_DBI ensures the objective regarding the management of the TSF data.

### *7.3.2 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 7.3.1 |
| 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, | Section 7.3.1 |

| | | |
|---|---|---|
| | 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, FDP_ACC.2/RV_Stack, FDP_ACF.1/RV_Stack, FMT_MSA.1/RV_Stack, FMT_MSA.2/RV_Stack, FMT_MSA.3/RV_Stack, FMT_SMF.1/RV_Stack, FDP_ACC.2/RV_Heap, FDP_ACF.1/RV_Heap, FMT_MSA.1/RV_Heap, FMT_MSA.2/RV_Heap, FMT_MSA.3/RV_Heap, FMT_SMF.1/RV_Heap, FDP_ACC.2/RV_Transient, FDP_ACF.1/RV_Transient, FMT_MSA.1/RV_Transient, FMT_MSA.2/RV_Transient, FMT_MSA.3/RV_Transient, FMT_SMF.1/RV_Transient | |
| 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, FDP_RIP.1/GlobalArray | Section 7.3.1 |
| O.GLOBAL_ARRAYS_INTEG | FDP_IFC.1/JCVM, FDP_IFF.1/JCVM | Section 7.3.1 |
| O.NATIVE | FDP_ACF.1/FIREWALL, FDP_ACF.1/JBox, FDP_ACC.2/JBox | Section 7.3.1 |
| O.OPERATE | FPT_RCV.4/SCP, FAU_ARP.1, FDP_ROL.1/FIREWALL, FIA_ATD.1/AID, FPT_FLS.1/ADEL, FPT_FLS.1, FPT_FLS.1/ODEL, FPT_FLS.1/Installer, FDP_ITC.2/Installer, FPT_RCV.3/Installer, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FPT_TDC.1, FIA_USB.1/AID, FPT_TST.1 | Section 7.3.1 |
| 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/OBJECTS, FDP_RIP.1/ADEL, FDP_RIP.1/ODEL, FDP_RIP.1/GlobalArray | Section 7.3.1 |

| | | |
|---|---|---|
| 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 7.3.1 |
| O.ARRAY_VIEWS_CONFID | FDP_IFF.1/JCVM, FDP_IFC.1/JCVM | Section 7.3.1 |
| O.ARRAY_VIEWS_INTEG | FDP_IFC.1/JCVM, FDP_IFF.1/JCVM | Section 7.3.1 |
| O.ALARM | FPT_FLS.1/Installer, FPT_FLS.1, FPT_FLS.1/ADEL, FPT_FLS.1/ODEL, FAU_ARP.1 | Section 7.3.1 |
| O.CIPHER | FCS_CKM.1, FCS_CKM.4, FCS_COP.1, FPR_UNO.1, FPR_UNO.1/USE_KEY, FCS_COP.1/PP, FCS_COP.1/CM, FCS_CKM.1/RSA, FCS_COP.1/RSA, FCS_COP.1/SHA3, FCS_COP.1/PACE_ENC, FCS_COP.1/PACE_MAC, FCS_COP.1/HMAC | Section 7.3.1 |
| O.RNG | FCS_RNG.1 | Section 7.3.1 |
| O.KEY-MNGT | FCS_CKM.1, 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/TRANSIENT, FDP_RIP.1/ADEL, FDP_SDI.2/DATA, FCS_COP.1/PP, FPR_UNO.1/USE_KEY, FDP_RIP.1/GlobalArray, FPT_TDC.1/CM, FCS_CKM.1/RSA, FCS_COP.1/RSA, FCS_COP.1/SHA3, FCS_COP.1/HMAC | Section 7.3.1 |
| O.PIN-MNGT | FDP_RIP.1/ODEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/ABORT, FDP_RIP.1/TRANSIENT, FPR_UNO.1, FDP_RIP.1/ADEL, FDP_ROL.1/FIREWALL, FDP_SDI.2/DATA, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FIA_AFL.1/PIN, FMT_MTD.2/GP_PIN, FMT_MTD.1/PIN, FIA_AFL.1/GP_PIN, FDP_RIP.1/GlobalArray | Section 7.3.1 |
| O.TRANSACTION | FDP_ROL.1/FIREWALL, FDP_RIP.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/ADEL, FDP_RIP.1/OBJECTS, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ODEL, FDP_RIP.1/GlobalArray | Section 7.3.1 |

| | | |
|---|---|---|
| O.OBJ-DELETION | FDP_RIP.1/ODEL, FPT_FLS.1/ODEL | Section 7.3.1 |
| O.DELETION | FDP_ACC.2/ADEL, FDP_ACF.1/ADEL, FDP_RIP.1/ADEL, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FPT_FLS.1/ADEL, FMT_SMR.1/ADEL, FPT_RCV.3/Installer | Section 7.3.1 |
| O.LOAD | FCO_NRO.2/CM, FDP_IFC.2/CM, FDP_IFF.1/CM, FDP_UIT.1/CM, FIA_UID.1/CM, FTP_ITC.1/CM | Section 7.3.1 |
| O.INSTALL | FDP_ITC.2/Installer, FPT_FLS.1/Installer, FPT_RCV.3/Installer | Section 7.3.1 |
| O.SCP.SUPPORT | FPT_RCV.4/SCP, FCS_CKM.1, FCS_CKM.4, FCS_COP.1, FCS_COP.1/CM, FCS_CKM.4/PP, FCS_COP.1/PP, FCS_CKM.1/RSA, FCS_COP.1/RSA, FCS_COP.1/SHA3, FCS_COP.1/PACE_ENC, FCS_COP.1/PACE_MAC, FCS_COP.1/HMAC | Section 7.3.1 |
| O.SCP.IC | FPT_PHP.3/SCP, FCS_RNG.1 | Section 7.3.1 |
| O.SCP.RECOVERY | FRU_FLT.1/SCP, FPT_RCV.4/SCP, FAU_ARP.1, FPT_FLS.1 | Section 7.3.1 |
| O.RESIDENT_APPLICATION | FDP_ACC.2/PP, FDP_ACF.1/PP, FDP_UCT.1/PP, FDP_ITC.1/PP, FIA_AFL.1/PP, FIA_UAU.1/PP, FMT_MSA.1/PP, FMT_SMF.1/PP, FMT_MOF.1/PP, FMT_SMR.2/PP, FMT_MSA.3/PP, FCS_CKM.4/PP, FMT_SMR.2/CM, FIA_UAU.1/CM, FIA_UAU.4/CardManu, FIA_UAU.7/CardManu, FIA_UAU.4/CardIssuer, FIA_UAU.7/CardIssuer, FDP_ACC.2/FlxC, FDP_ACF.1/FlxC, FMT_MOF.1/FlxC, FPT_FLS.1/FlxC, FDP_RIP.1/FlxC | Section 7.3.1 |
| O.CARD_MANAGEMENT | FDP_ACC.2/ADEL, FDP_ACF.1/ADEL, FDP_RIP.1/ADEL, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FPT_FLS.1/ADEL, FDP_ITC.2/Installer, FPT_FLS.1/Installer, FPT_RCV.3/Installer, FDP_UIT.1/CM, FDP_ROL.1/FIREWALL, FPT_FLS.1/ODEL, FIA_AFL.1/CM, FPT_TST.1, FIA_UID.1/CM, FDP_IFF.1/CM, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.2/PP, FMT_SMF.1/PP, FTP_ITC.1/CM, FMT_SMR.2/CM, FDP_IFC.2/CM, FCO_NRO.2/CM_DAP, FIA_UAU.7/CardIssuer, | Section 7.3.1 |

| | | |
|---|---|---|
| | FPR_UNO.1/Key_CM, FIA_UAU.4/CardIssuer, FPT_TDC.1/CM, FIA_UAU.4/CardManu, FIA_UAU.7/CardManu, FMT_SMF.1/CM, FMT_SMR.1/CM, FIA_UAU.1/CM, FCO_NRO.2/CM | |
| O.SECURE_COMPARE | FDP_SDI.2/DATA | Section 7.3.1 |
| O.PATCH_LOADING | FDP_ACC.2/PP, FDP_ACF.1/PP, FIA_UAU.1/PP, FIA_UID.1/PP, FCS_COP.1/PP, FTP_ITC.1/CM, FCS_COP.1/CM, FDP_UIT.1/PP, FDP_ITC.1/PP, FDP_UCT.1/PP, FMT_MOF.1/PP, FCS_CKM.4/PP, FAU_STG.2, FIA_AFL.1/PP, FPT_TST.1 | Section 7.3.1 |
| O.JBox_FW | FDP_ACC.2/JBox, FDP_ACF.1/JBox, FMT_MSA.1/JBox, FMT_MSA.3/JBox, FMT_SMF.1/JBox | Section 7.3.1 |
| O.FLEXICODE | FMT_SMF.1/FlxC, FDP_ACC.2/FlxC, FDP_ACF.1/FlxC, FMT_MSA.1/FlxC, FMT_MSA.2/FlxC, FMT_MSA.3/FlxC, FMT_MSA.4/FlxC, FMT_MOF.1/FlxC, FPT_FLS.1/FlxC, FDP_RIP.1/FlxC, FPT_TST.1 | Section 7.3.1 |
| O.DUMP_PERSO | FCS_COP.1/PP, FDP_ITC.1/PP, FIA_UAU.1/PP, FIA_UID.1/PP, FMT_SMR.2/CM, FDP_UIT.1/PP, FDP_UCT.1/PP, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FIA_AFL.1/CM, FTP_ITC.1/CM, FPT_TST.1, FDP_ACC.2/PP, FCS_COP.1/CM, FDP_SDI.2/DATA | Section 7.3.1 |
| O.CLFDB_DECIPHER | FCS_COP.1/CM, FDP_ITC.2/Installer, FCS_CKM.4, FPR_UNO.1/USE_KEY, FPR_UNO.1 | Section 7.3.1 |
| O.MTC-CTR-MNGT | FDP_SDI.2/MONOTONIC_COUNTER, FDP_RIP.1/OBJECTS, FDP_RIP.1/ADEL, FDP_RIP.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/TRANSIENT, FDP_ROL.1/FIREWALL, FDP_RIP.1/GlobalArray, FDP_RIP.1/bArray, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FDP_RIP.1/ODEL | Section 7.3.1 |
| O.PACE_Data_Integrity | FCS_CKM.1/DH_PACE, FCS_CKM.4/DH_PACE, FCS_COP.1/PACE_MAC, FCS_RND.1/PACE, FDP_RIP.1/PACE, FIA_UID.1/PACE, | Section 7.3.1 |

| | | |
|---|---|---|
| | FIA_UAU.1/PACE, FIA_UAU.4/PACE, FIA_UAU.5/PACE, FTP_ITC.1/PACE, FMT_SMR.1/PACE, FMT_SMF.1/PACE, FMT_MTD.1/KEY_READ, FPT_PHP.3/SCP, FIA_AFL.1/PACE, FIA_UID.1/PACE_CAM, FIA_UAU.1/PACE_CAM, FIA_UAU.4/PACE_CAM, FIA_UAU.5/PACE_CAM, FMT_MTD.1/PACE_CAM_KEY_READ, FMT_MTD.1/PACE_CAM_KEY_WRITE, FCS_CKM.4, FIA_UAU.6/PACE, FIA_UAU.6/PACE_CAM, FCS_CKM.1/DH_PACE_DH | |
| O.PACE_Data_Authenticity | FCS_CKM.1/DH_PACE, FCS_CKM.4/DH_PACE, FCS_COP.1/PACE_MAC, FDP_RIP.1/PACE, FIA_AFL.1/PACE, FIA_UID.1/PACE, FIA_UAU.1/PACE, FIA_UAU.4/PACE, FIA_UAU.5/PACE, FIA_UAU.6/PACE, FTP_ITC.1/PACE, FMT_SMR.1/PACE, FMT_SMF.1/PACE, FMT_MTD.1/KEY_READ, FCS_RND.1/PACE, FIA_UAU.6/PACE_CAM, FIA_UAU.5/PACE_CAM, FIA_UAU.1/PACE_CAM, FIA_UAU.4/PACE_CAM, FMT_MTD.1/PACE_CAM_KEY_READ, FCS_CKM.4, FIA_UID.1/PACE_CAM, FCS_CKM.1/DH_PACE_DH | Section 7.3.1 |
| O.PACE_Data_Confidentiality | FCS_CKM.1/DH_PACE, FCS_CKM.4/DH_PACE, FCS_COP.1/PACE_ENC, FCS_RND.1/PACE, FDP_RIP.1/PACE, FIA_AFL.1/PACE, FIA_UID.1/PACE, FIA_UAU.1/PACE, FIA_UAU.4/PACE, FIA_UAU.5/PACE, FIA_UAU.6/PACE, FTP_ITC.1/PACE, FMT_SMR.1/PACE, FMT_SMF.1/PACE, FMT_MTD.1/KEY_READ, FCS_COP.1/PACE_MAC, FIA_UAU.6/PACE_CAM, FIA_UAU.5/PACE_CAM, FIA_UAU.1/PACE_CAM, FIA_UAU.4/PACE_CAM, FIA_UID.1/PACE_CAM, FMT_MTD.1/PACE_CAM_KEY_READ, FCS_CKM.4, FCS_CKM.1/DH_PACE_DH | Section 7.3.1 |
| O.PACE_Prot_Abuse-Func | FMT_LIM.1/PACE_Perso, FMT_LIM.2/PACE_Perso | Section 7.3.1 |

| | | |
|---|---|---|
| O.PACE_Prot_Inf_Leak | FPT_FLS.1, FPT_PHP.3/SCP, FPT_TST.1, FPT_EMS.1/PACE | Section 7.3.1 |
| O.PACE_Prot_Phys-Tamper | FPT_PHP.3/SCP | Section 7.3.1 |
| O.PACE_Prot_Malfunction | FPT_TST.1, FPT_FLS.1 | Section 7.3.1 |
| O.PACE_Identification | FMT_SMR.1/PACE, FMT_SMF.1/PACE, FMT_MTD.1/INI_ENA, FMT_MTD.1/INI_DIS | Section 7.3.1 |
| O.PACE_AC_Pers | FMT_MTD.1/INI_ENA, FMT_MTD.1/INI_DIS, FMT_MTD.1/KEY_READ, FMT_SMR.1/PACE, FMT_SMF.1/PACE, FIA_UID.1/PACE_CAM, FIA_UAU.1/PACE_CAM, FIA_UAU.4/PACE_CAM, FIA_UAU.5/PACE_CAM, FMT_MTD.1/PACE_CAM_KEY_READ, FMT_MTD.1/PACE_CAM_KEY_WRITE | Section 7.3.1 |
| O.SENSITIVE_ARRAYS_INTEG | FDP_SDI.2/ARRAY | Section 7.3.1 |
| 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, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FPR_UNO.1, FDP_ROL.1/FIREWALL, FDP_SDI.2/DATA, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FIA_AFL.1/CM, FIA_AFL.1/BIO, FMT_MTD.1/BIO | Section 7.3.1 |
| O.DBI-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, FDP_RIP.1/ADEL, FDP_RIP.1/TRANSIENT, FPR_UNO.1, FDP_ROL.1/FIREWALL, FDP_SDI.2/DATA, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FIA_AFL.1/CM, FMT_MTD.1/Activate_DBI, FMT_MTD.1/Deactivate_DBI | Section 7.3.1 |

**Table 11 Security Objectives and SFRs - Coverage**

| Security Functional Requirements | Security Objectives | Rationale |
|---|---|---|
| FCS_CKM.1 | O.CIPHER, O.KEY-MNGT, O.SCP.SUPPORT | |
| FCS_CKM.4 | O.CIPHER, O.KEY-MNGT, O.SCP.SUPPORT, O.CLFDB_DECIPHER, O.PACE_Data_Integrity, | |

| | | |
|---|---|---|
| | O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FCS_COP.1 | O.CIPHER, O.KEY-MNGT, O.SCP.SUPPORT | |
| FDP_RIP.1/ABORT | O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.MTC-CTR-MNGT, O.BIO-MNGT, O.DBI-MNGT | |
| FDP_RIP.1/APDU | O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.MTC-CTR-MNGT, O.BIO-MNGT, O.DBI-MNGT | |
| FDP_RIP.1/GlobalArray | O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.MTC-CTR-MNGT | |
| FDP_RIP.1/bArray | O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.MTC-CTR-MNGT, O.BIO-MNGT, O.DBI-MNGT | |
| FDP_RIP.1/KEYS | O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.BIO-MNGT, O.DBI-MNGT | |
| FDP_RIP.1/TRANSIENT | O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.MTC-CTR-MNGT, O.BIO-MNGT, O.DBI-MNGT | |
| FDP_ROL.1/FIREWALL | O.OPERATE, O.RESOURCES, O.PIN-MNGT, O.TRANSACTION, O.CARD_MANAGEMENT, O.MTC-CTR-MNGT, O.BIO-MNGT, O.DBI-MNGT | |
| FCS_RNG.1 | O.RNG, O.SCP.IC | |
| FDP_ACC.2/FIREWALL | O.FIREWALL, O.OPERATE, O.PIN-MNGT, O.DUMP_PERSO, O.MTC-CTR-MNGT, O.BIO-MNGT, O.DBI-MNGT | |
| FDP_ACF.1/FIREWALL | O.FIREWALL, O.NATIVE, O.OPERATE, O.PIN-MNGT, O.DUMP_PERSO, O.MTC-CTR-MNGT, O.BIO-MNGT, O.DBI-MNGT | |
| FDP_IFC.1/JCVM | O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, | |

| | | |
|---|---|---|
| | O.ARRAY_VIEWS_CONFID, O.ARRAY_VIEWS_INTEG | |
| FDP_IFF.1/JCVM | O.FIREWALL, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.ARRAY_VIEWS_CONFID, O.ARRAY_VIEWS_INTEG | |
| FDP_RIP.1/OBJECTS | O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.MTC-CTR-MNGT, O.BIO-MNGT, O.DBI-MNGT | |
| FMT_MSA.1/JCRE | O.SID, O.FIREWALL | |
| FMT_MSA.1/JCVM | O.SID, O.FIREWALL | |
| FMT_MSA.2/FIREWALL_JCVM | O.FIREWALL | |
| FMT_MSA.3/FIREWALL | O.SID, O.FIREWALL | |
| FMT_MSA.3/JCVM | O.SID, O.FIREWALL | |
| FMT_SMF.1 | O.FIREWALL, O.RESOURCES | |
| FMT_SMR.1 | O.FIREWALL, O.RESOURCES | |
| FAU_ARP.1 | O.OPERATE, O.RESOURCES, O.ALARM, O.SCP.RECOVERY | |
| FDP_SDI.2/DATA | O.KEY-MNGT, O.PIN-MNGT, O.SECURE_COMPARE, O.DUMP_PERSO, O.BIO-MNGT, O.DBI-MNGT | |
| FPR_UNO.1 | O.CIPHER, O.KEY-MNGT, O.PIN-MNGT, O.CLFDB_DECIPHER, O.BIO-MNGT, O.DBI-MNGT | |
| FPT_FLS.1 | O.OPERATE, O.RESOURCES, O.ALARM, O.SCP.RECOVERY, O.PACE_Prot_Inf_Leak, O.PACE_Prot_Malfunction | |
| 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, O.CARD_MANAGEMENT, O.CLFDB_DECIPHER | |

| | | |
|---|---|---|
| FMT_SMR.1/Installer | O.FIREWALL, O.RESOURCES | |
| FPT_FLS.1/Installer | O.OPERATE, O.RESOURCES, O.ALARM, O.INSTALL, O.CARD_MANAGEMENT | |
| FPT_RCV.3/Installer | O.OPERATE, O.RESOURCES, O.DELETION, O.INSTALL, O.CARD_MANAGEMENT | |
| FDP_ACC.2/ADEL | O.DELETION, O.CARD_MANAGEMENT | |
| FDP_ACF.1/ADEL | O.DELETION, O.CARD_MANAGEMENT | |
| FDP_RIP.1/ADEL | O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.DELETION, O.CARD_MANAGEMENT, O.MTC-CTR-MNGT, O.BIO-MNGT, O.DBI-MNGT | |
| FMT_MSA.1/ADEL | O.SID, O.FIREWALL, O.DELETION, O.CARD_MANAGEMENT | |
| FMT_MSA.3/ADEL | O.SID, O.FIREWALL, O.DELETION, O.CARD_MANAGEMENT | |
| FMT_SMF.1/ADEL | O.SID, O.FIREWALL, O.RESOURCES | |
| FMT_SMR.1/ADEL | O.FIREWALL, O.RESOURCES, O.DELETION, O.CARD_MANAGEMENT | |
| FPT_FLS.1/ADEL | O.OPERATE, O.RESOURCES, O.ALARM, O.DELETION, O.CARD_MANAGEMENT | |
| FDP_RIP.1/ODEL | O.GLOBAL_ARRAYS_CONFID, O.REALLOCATION, O.KEY-MNGT, O.PIN-MNGT, O.TRANSACTION, O.OBJ-DELETION, O.MTC-CTR-MNGT, O.BIO-MNGT, O.DBI-MNGT | |
| FPT_FLS.1/ODEL | O.OPERATE, O.RESOURCES, O.ALARM, O.OBJ-DELETION, O.CARD_MANAGEMENT | |
| FCO_NRO.2/CM | O.LOAD, O.CARD_MANAGEMENT | |
| FDP_IFC.2/CM | O.LOAD, O.CARD_MANAGEMENT | |
| FDP_IFF.1/CM | O.LOAD, O.CARD_MANAGEMENT | |
| FDP_UIT.1/CM | O.LOAD, O.CARD_MANAGEMENT | |
| FIA_UID.1/CM | O.LOAD, O.CARD_MANAGEMENT | |
| FMT_MSA.1/CM | O.SID, O.FIREWALL, O.CARD_MANAGEMENT | |
| FMT_MSA.3/CM | O.SID, O.FIREWALL, O.CARD_MANAGEMENT | |

| | | |
|---|---|---|
| FMT_SMF.1/CM | O.SID, O.FIREWALL, O.RESOURCES, O.CARD_MANAGEMENT | |
| FMT_SMR.1/CM | O.FIREWALL, O.RESOURCES, O.CARD_MANAGEMENT | |
| FTP_ITC.1/CM | O.LOAD, O.CARD_MANAGEMENT, O.PATCH_LOADING, O.DUMP_PERSO | |
| FPT_TST.1 | O.OPERATE, O.CARD_MANAGEMENT, O.PATCH_LOADING, O.FLEXICODE, O.DUMP_PERSO, O.PACE_Prot_Inf_Leak, O.PACE_Prot_Malfunction | |
| FCO_NRO.2/CM_DAP | O.CARD_MANAGEMENT | |
| FIA_AFL.1/CM | O.CARD_MANAGEMENT, O.DUMP_PERSO, O.BIO-MNGT, O.DBI-MNGT | |
| FIA_UAU.1/CM | O.RESIDENT_APPLICATION, O.CARD_MANAGEMENT | |
| FIA_UAU.4/CardIssuer | O.RESIDENT_APPLICATION, O.CARD_MANAGEMENT | |
| FIA_UAU.7/CardIssuer | O.RESIDENT_APPLICATION, O.CARD_MANAGEMENT | |
| FPR_UNO.1/Key_CM | O.CARD_MANAGEMENT | |
| FPT_TDC.1/CM | O.KEY-MNGT, O.CARD_MANAGEMENT | |
| FMT_SMR.2/CM | O.RESIDENT_APPLICATION, O.CARD_MANAGEMENT, O.DUMP_PERSO | |
| FCS_COP.1/CM | O.CIPHER, O.SCP.SUPPORT, O.PATCH_LOADING, O.DUMP_PERSO, O.CLFDB_DECIPHER | |
| FDP_ACC.2/PP | O.RESIDENT_APPLICATION, O.PATCH_LOADING, O.DUMP_PERSO | |
| FDP_ACF.1/PP | O.RESIDENT_APPLICATION, O.PATCH_LOADING | |
| FDP_UCT.1/PP | O.RESIDENT_APPLICATION, O.PATCH_LOADING, O.DUMP_PERSO | |
| FDP_ITC.1/PP | O.RESIDENT_APPLICATION, O.PATCH_LOADING, O.DUMP_PERSO | |
| FIA_AFL.1/PP | O.RESIDENT_APPLICATION, O.PATCH_LOADING | |

| | | |
|---|---|---|
| FIA_UAU.1/PP | O.RESIDENT_APPLICATION, O.PATCH_LOADING, O.DUMP_PERSO | |
| FIA_UID.1/PP | O.PATCH_LOADING, O.DUMP_PERSO | |
| FMT_MSA.1/PP | O.RESIDENT_APPLICATION | |
| FMT_SMF.1/PP | O.RESIDENT_APPLICATION, O.CARD_MANAGEMENT | |
| FIA_UAU.4/CardManu | O.RESIDENT_APPLICATION, O.CARD_MANAGEMENT | |
| FIA_UAU.7/CardManu | O.RESIDENT_APPLICATION, O.CARD_MANAGEMENT | |
| FMT_MOF.1/PP | O.RESIDENT_APPLICATION, O.PATCH_LOADING | |
| FMT_SMR.2/PP | O.RESIDENT_APPLICATION, O.CARD_MANAGEMENT | |
| FMT_MSA.3/PP | O.RESIDENT_APPLICATION | |
| FCS_COP.1/PP | O.CIPHER, O.KEY-MNGT, O.SCP.SUPPORT, O.PATCH_LOADING, O.DUMP_PERSO | |
| FCS_CKM.4/PP | O.SCP.SUPPORT, O.RESIDENT_APPLICATION, O.PATCH_LOADING | |
| FDP_UIT.1/PP | O.PATCH_LOADING, O.DUMP_PERSO | |
| FAU_STG.2 | O.PATCH_LOADING | |
| FPT_PHP.3/SCP | O.SCP.IC, O.PACE_Data_Integrity, O.PACE_Prot_Inf_Leak, O.PACE_Prot_Phys-Tamper | |
| FPT_RCV.4/SCP | O.OPERATE, O.SCP.SUPPORT, O.SCP.RECOVERY | |
| FRU_FLT.1/SCP | O.SCP.RECOVERY | |
| FPR_UNO.1/USE_KEY | O.CIPHER, O.KEY-MNGT, O.CLFDB_DECIPHER | |
| FIA_AFL.1/PIN | O.PIN-MNGT | |
| FMT_MTD.2/GP_PIN | O.PIN-MNGT | |
| FMT_MTD.1/PIN | O.PIN-MNGT | |
| FIA_AFL.1/GP_PIN | O.PIN-MNGT | |
| FDP_ACC.2/RV_Stack | O.FIREWALL | |
| FDP_ACF.1/RV_Stack | O.FIREWALL | |
| FMT_MSA.1/RV_Stack | O.FIREWALL | |

| | | |
|---|---|---|
| FMT_MSA.2/RV_Stack | O.FIREWALL | |
| FMT_MSA.3/RV_Stack | O.FIREWALL | |
| FMT_SMF.1/RV_Stack | O.FIREWALL | |
| FDP_ACC.2/RV_Heap | O.FIREWALL | |
| FDP_ACF.1/RV_Heap | O.FIREWALL | |
| FMT_MSA.1/RV_Heap | O.FIREWALL | |
| FMT_MSA.2/RV_Heap | O.FIREWALL | |
| FMT_MSA.3/RV_Heap | O.FIREWALL | |
| FMT_SMF.1/RV_Heap | O.FIREWALL | |
| FDP_ACC.2/RV_Transient | O.FIREWALL | |
| FDP_ACF.1/RV_Transient | O.FIREWALL | |
| FMT_MSA.1/RV_Transient | O.FIREWALL | |
| FMT_MSA.2/RV_Transient | O.FIREWALL | |
| FMT_MSA.3/RV_Transient | O.FIREWALL | |
| FMT_SMF.1/RV_Transient | O.FIREWALL | |
| FDP_ACC.2/JBox | O.NATIVE, O.JBox_FW | |
| FDP_ACF.1/JBox | O.NATIVE, O.JBox_FW | |
| FMT_MSA.1/JBox | O.JBox_FW | |
| FMT_MSA.3/JBox | O.JBox_FW | |
| FMT_SMF.1/JBox | O.JBox_FW | |
| FMT_MTD.1/Activate_DBI | O.DBI-MNGT | |
| FMT_MTD.1/Deactivate_DBI | O.DBI-MNGT | |
| FMT_MOF.1/FlxC | O.RESIDENT_APPLICATION, O.FLEXICODE | |
| FMT_SMF.1/FlxC | O.FLEXICODE | |
| FDP_ACC.2/FlxC | O.RESIDENT_APPLICATION, O.FLEXICODE | |
| FDP_ACF.1/FlxC | O.RESIDENT_APPLICATION, O.FLEXICODE | |
| FMT_MSA.1/FlxC | O.FLEXICODE | |
| FMT_MSA.2/FlxC | O.FLEXICODE | |
| FMT_MSA.3/FlxC | O.FLEXICODE | |
| FMT_MSA.4/FlxC | O.FLEXICODE | |

| | | |
|---|---|---|
| FPT_FLS.1/FlxC | O.RESIDENT_APPLICATION, O.FLEXICODE | |
| FDP_RIP.1/FlxC | O.RESIDENT_APPLICATION, O.FLEXICODE | |
| FDP_SDI.2/MONOTONIC_COUNTER | O.MTC-CTR-MNGT | |
| FDP_SDI.2/ARRAY | O.SENSITIVE_ARRAYS_INTEG | |
| FCS_CKM.1/DH_PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FCS_COP.1/PACE_ENC | O.CIPHER, O.SCP.SUPPORT, O.PACE_Data_Confidentiality | |
| FCS_COP.1/PACE_MAC | O.CIPHER, O.SCP.SUPPORT, O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FCS_CKM.4/DH_PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FCS_RND.1/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FIA_AFL.1/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FIA_UID.1/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FIA_UAU.1/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FIA_UAU.4/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FIA_UAU.5/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FIA_UAU.6/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FTP_ITC.1/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |

| | | |
|---|---|---|
| FMT_SMR.1/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality, O.PACE_Identification, O.PACE_AC_Pers | |
| FMT_SMF.1/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality, O.PACE_Identification, O.PACE_AC_Pers | |
| FMT_LIM.1/PACE_Perso | O.PACE_Prot_Abuse-Func | |
| FMT_LIM.2/PACE_Perso | O.PACE_Prot_Abuse-Func | |
| FMT_MTD.1/INI_ENA | O.PACE_Identification, O.PACE_AC_Pers | |
| FMT_MTD.1/INI_DIS | O.PACE_Identification, O.PACE_AC_Pers | |
| FMT_MTD.1/KEY_READ | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality, O.PACE_AC_Pers | |
| FDP_RIP.1/PACE | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FPT_EMS.1/PACE | O.PACE_Prot_Inf_Leak | |
| FIA_UID.1/PACE_CAM | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality, O.PACE_AC_Pers | |
| FIA_UAU.1/PACE_CAM | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality, O.PACE_AC_Pers | |
| FIA_UAU.4/PACE_CAM | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality, O.PACE_AC_Pers | |
| FIA_UAU.5/PACE_CAM | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality, O.PACE_AC_Pers | |
| FIA_UAU.6/PACE_CAM | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FMT_MTD.1/PACE_CAM_KEY_READ | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, | |

| | | |
|---|---|---|
| | O.PACE_Data_Confidentiality, O.PACE_AC_Pers | |
| FMT_MTD.1/PACE_CAM_KEY_WRITE | O.PACE_Data_Integrity, O.PACE_AC_Pers | |
| FCS_CKM.1/RSA | O.CIPHER, O.KEY-MNGT, O.SCP.SUPPORT | |
| FCS_COP.1/RSA | O.CIPHER, O.KEY-MNGT, O.SCP.SUPPORT | |
| FCS_COP.1/SHA3 | O.CIPHER, O.KEY-MNGT, O.SCP.SUPPORT | |
| FIA_AFL.1/BIO | O.BIO-MNGT | |
| FMT_MTD.1/BIO | O.BIO-MNGT | |
| FCS_CKM.1/DH_PACE_DH | O.PACE_Data_Integrity, O.PACE_Data_Authenticity, O.PACE_Data_Confidentiality | |
| FCS_COP.1/HMAC | O.CIPHER, O.KEY-MNGT, O.SCP.SUPPORT | |

**Table 12 SFRs and Security Objectives**

### *7.3.3 Dependencies*

#### 7.3.3.1 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) | FPT_TDC.1, FDP_IFC.2/CM, FTP_ITC.1/CM |
| 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 | |
| FDP_ACC.2/JBox | (FDP_ACF.1) | FDP_ACF.1/JBox |
| FDP_ACF.1/JBox | (FDP_ACC.1) and (FMT_MSA.3) | FDP_ACC.2/JBox, FMT_MSA.3/JBox |
| FMT_MSA.1/JBox | (FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1) | FDP_ACC.2/JBox, FMT_SMF.1/JBox, FMT_SMR.1 |
| FMT_MSA.3/JBox | (FMT_MSA.1) and (FMT_SMR.1) | FMT_MSA.1/JBox, FMT_SMR.1 |
| FMT_SMF.1/JBox | No Dependencies | |
| FMT_MOF.1/FlxC | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMF.1/FlxC, FMT_SMR.1/CM, FMT_SMR.2/PP |
| FMT_SMF.1/FlxC | No Dependencies | |
| FDP_ACC.2/FlxC | (FDP_ACF.1) | FDP_ACF.1/FlxC, FDP_ACF.1/PP |
| FDP_ACF.1/FlxC | (FDP_ACC.1) and (FMT_MSA.3) | FDP_ACC.2/FlxC, FDP_ACC.2/PP, FMT_MSA.3/PP |
| FMT_MSA.1/FlxC | (FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMF.1/FlxC, FDP_ACC.2/FlxC, FMT_SMR.1/CM, FMT_SMR.2/PP |
| FMT_MSA.2/FlxC | (FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1) | FDP_ACC.2/FlxC, FMT_MSA.1/FlxC, FMT_SMR.1/CM, FMT_SMR.2/PP |
| FMT_MSA.3/FlxC | (FMT_MSA.1) and (FMT_SMR.1) | FMT_MSA.1/FlxC, FMT_SMR.1/CM, FMT_SMR.2/PP |
| FMT_MSA.4/FlxC | (FDP_ACC.1 or FDP_IFC.1) | FDP_ACC.2/FlxC |
| FPT_FLS.1/FlxC | No Dependencies | |
| FDP_RIP.1/FlxC | No Dependencies | |

| | | |
|---|---|---|
| FDP_SDI.2/MONOTONIC_COUNTER | No Dependencies | |
| FDP_SDI.2/ARRAY | No Dependencies | |
| FIA_UID.1/PACE_CAM | No Dependencies | |
| FIA_UAU.1/PACE_CAM | (FIA_UID.1) | FIA_UID.1/PACE_CAM |
| FIA_UAU.4/PACE_CAM | No Dependencies | |
| FIA_UAU.5/PACE_CAM | No Dependencies | |
| FIA_UAU.6/PACE_CAM | No Dependencies | |
| FMT_MTD.1/PACE_CAM_KEY_READ | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMR.1/PACE, FMT_SMF.1/PACE |
| FMT_MTD.1/PACE_CAM_KEY_WRITE | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMR.1/PACE, FMT_SMF.1/PACE |
| FCS_CKM.1/RSA | (FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4) | FCS_COP.1/RSA, FCS_CKM.4 |
| FCS_COP.1/RSA | (FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4) | FCS_CKM.1/RSA, FCS_CKM.4 |
| FCS_COP.1/SHA3 | (FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4) | FCS_CKM.1, FCS_CKM.4 |
| FIA_AFL.1/BIO | (FIA_UAU.1) | FIA_UAU.1/PP |
| FMT_MTD.1/BIO | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMF.1/CM, FMT_SMR.2/CM |
| FCS_CKM.1/DH_PACE_DH | (FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4) | FCS_CKM.4, FCS_COP.1/PACE_ENC, FCS_COP.1/PACE_MAC |
| FCS_COP.1/HMAC | (FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4) | FCS_CKM.1, FCS_CKM.4 |
| FCS_CKM.1 | (FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4) | FCS_CKM.4, FCS_COP.1 |
| 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 | FCS_CKM.1, FCS_CKM.4 |

| | | |
|---|---|---|
| | FDP_ITC.2) and (FCS_CKM.4) | |
| FDP_RIP.1/ABORT | No Dependencies | |
| FDP_RIP.1/APDU | No Dependencies | |
| FDP_RIP.1/GlobalArray | 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 |
| FCS_RNG.1 | 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 |
| 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 |
| FAU_ARP.1 | (FAU_SAA.1) | |
| FDP_SDI.2/DATA | No Dependencies | |

| | | |
|---|---|---|
| FPR_UNO.1 | No 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 |
| FCO_NRO.2/CM | (FIA_UID.1) | FIA_UID.1/CM |
| FDP_IFC.2/CM | (FDP_IFF.1) | FDP_IFF.1/CM |
| FDP_IFF.1/CM | (FDP_IFC.1) and (FMT_MSA.3) | FDP_IFC.2/CM, FMT_MSA.3/CM |
| FDP_UIT.1/CM | (FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1) | FDP_IFC.2/CM, FTP_ITC.1/CM |
| FIA_UID.1/CM | No Dependencies | |
| FMT_MSA.1/CM | (FDP_ACC.1 or FDP_IFC.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_TST.1 | No Dependencies | |
| FCO_NRO.2/CM_DAP | (FIA_UID.1) | FIA_UID.1/PP |
| FIA_AFL.1/CM | (FIA_UAU.1) | FIA_UAU.1/CM |
| FIA_UAU.1/CM | (FIA_UID.1) | FIA_UID.1/CM |
| FIA_UAU.4/CardIssuer | No Dependencies | |
| FIA_UAU.7/CardIssuer | (FIA_UAU.1) | FIA_UAU.1/CM |
| FPR_UNO.1/Key_CM | No Dependencies | |
| FPT_TDC.1/CM | No Dependencies | |
| FMT_SMR.2/CM | (FIA_UID.1) | FIA_UID.1/PP |

| | | |
|---|---|---|
| FCS_COP.1/CM | (FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4) | FCS_CKM.1, FCS_CKM.4 |
| FDP_ACC.2/PP | (FDP_ACF.1) | FDP_ACF.1/PP |
| FDP_ACF.1/PP | (FDP_ACC.1) and (FMT_MSA.3) | FDP_ACC.2/PP, FMT_MSA.3/PP |
| FDP_UCT.1/PP | (FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1) | FTP_ITC.1/CM, FDP_ACC.2/PP |
| FDP_ITC.1/PP | (FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.3) | FDP_ACC.2/PP, FMT_MSA.3/PP |
| FIA_AFL.1/PP | (FIA_UAU.1) | FIA_UAU.1/PP |
| FIA_UAU.1/PP | (FIA_UID.1) | FIA_UID.1/PP |
| FIA_UID.1/PP | No Dependencies | |
| FMT_MSA.1/PP | (FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1) | FDP_ACC.2/PP, FMT_SMF.1/PP, FMT_SMR.2/PP |
| FMT_SMF.1/PP | No Dependencies | |
| FIA_UAU.4/CardManu | No Dependencies | |
| FIA_UAU.7/CardManu | (FIA_UAU.1) | FIA_UAU.1/PP |
| FMT_MOF.1/PP | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMF.1/PP, FMT_SMR.2/PP |
| FMT_SMR.2/PP | (FIA_UID.1) | FIA_UID.1/PP |
| FMT_MSA.3/PP | (FMT_MSA.1) and (FMT_SMR.1) | FMT_MSA.1/PP, FMT_SMR.2/PP |
| FCS_COP.1/PP | (FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4) | FDP_ITC.1/PP, FCS_CKM.4/PP |
| FCS_CKM.4/PP | (FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) | FDP_ITC.1/PP |
| FDP_UIT.1/PP | (FDP_ACC.1 or FDP_IFC.1) and (FTP_ITC.1 or FTP_TRP.1) | FTP_ITC.1/CM, FDP_ACC.2/PP |

| | | |
|---|---|---|
| FAU_STG.2 | (FAU_GEN.1) | |
| FPT_PHP.3/SCP | No Dependencies | |
| FPT_RCV.4/SCP | No Dependencies | |
| FRU_FLT.1/SCP | (FPT_FLS.1) | FPT_FLS.1 |
| FPR_UNO.1/USE_KEY | No Dependencies | |
| FIA_AFL.1/PIN | (FIA_UAU.1) | |
| FMT_MTD.2/GP_PIN | (FMT_MTD.1) and (FMT_SMR.1) | FMT_SMR.1, FMT_MTD.1/PIN |
| FMT_MTD.1/PIN | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMF.1, FMT_SMR.1 |
| FIA_AFL.1/GP_PIN | (FIA_UAU.1) | |
| FMT_MTD.1/Activate_DBI | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMF.1, FMT_SMF.1/CM, FMT_SMR.1/CM, FMT_SMR.2/CM, FMT_SMR.1/PACE |
| FMT_MTD.1/Deactivate_DBI | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMF.1, FMT_SMF.1/CM, FMT_SMR.1/CM, FMT_SMR.2/CM, FMT_SMR.1/PACE |
| FCS_CKM.1/DH_PACE | (FCS_CKM.2 or FCS_COP.1) and (FCS_CKM.4) | FCS_CKM.4, FCS_COP.1/PACE_ENC, FCS_COP.1/PACE_MAC |
| FCS_COP.1/PACE_ENC | (FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4) | FCS_CKM.1/DH_PACE, FCS_CKM.4/DH_PACE |
| FCS_COP.1/PACE_MAC | (FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) and (FCS_CKM.4) | FCS_CKM.1/DH_PACE, FCS_CKM.4/DH_PACE |
| FCS_CKM.4/DH_PACE | (FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2) | FCS_CKM.1/DH_PACE |
| FCS_RND.1/PACE | No Dependencies | |
| FIA_AFL.1/PACE | (FIA_UAU.1) | FIA_UAU.1/PACE |
| FIA_UID.1/PACE | No Dependencies | |
| FIA_UAU.1/PACE | (FIA_UID.1) | FIA_UID.1/PACE |
| FIA_UAU.4/PACE | No Dependencies | |
| FIA_UAU.5/PACE | No Dependencies | |

| | | |
|---|---|---|
| FIA_UAU.6/PACE | No Dependencies | |
| FTP_ITC.1/PACE | No Dependencies | |
| FMT_SMR.1/PACE | (FIA_UID.1) | FIA_UID.1/PACE_CAM, FIA_UID.1/PACE |
| FMT_SMF.1/PACE | No Dependencies | |
| FMT_LIM.1/PACE_Perso | No Dependencies | |
| FMT_LIM.2/PACE_Perso | No Dependencies | |
| FMT_MTD.1/INI_ENA | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMR.1/PACE, FMT_SMF.1/PACE |
| FMT_MTD.1/INI_DIS | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMR.1/PACE, FMT_SMF.1/PACE |
| FMT_MTD.1/KEY_READ | (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMR.1/PACE, FMT_SMF.1/PACE |
| FDP_RIP.1/PACE | No Dependencies | |
| FPT_EMS.1/PACE | No Dependencies | |
| FDP_ACC.2/RV_Stack | (FDP_ACF.1) | FDP_ACF.1/RV_Stack |
| FDP_ACF.1/RV_Stack | (FDP_ACC.1) and (FMT_MSA.3) | FDP_ACC.2/RV_Stack, FMT_MSA.3/RV_Stack |
| FMT_MSA.1/RV_Stack | (FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMR.1, FDP_ACC.2/RV_Stack, FMT_SMF.1/RV_Stack |
| FMT_MSA.2/RV_Stack | (FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1) | FMT_SMR.1, FDP_ACC.2/RV_Stack, FMT_MSA.1/RV_Stack |
| FMT_MSA.3/RV_Stack | (FMT_MSA.1) and (FMT_SMR.1) | FMT_SMR.1, FMT_MSA.1/RV_Stack |
| FMT_SMF.1/RV_Stack | No Dependencies | |
| FDP_ACC.2/RV_Heap | (FDP_ACF.1) | FDP_ACF.1/RV_Heap |
| FDP_ACF.1/RV_Heap | (FDP_ACC.1) and (FMT_MSA.3) | FDP_ACC.2/RV_Heap, FMT_MSA.3/RV_Heap |
| FMT_MSA.1/RV_Heap | (FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMR.1, FDP_ACC.2/RV_Heap, FMT_SMF.1/RV_Heap |
| FMT_MSA.2/RV_Heap | (FDP_ACC.1 or FDP_IFC.1) and | FMT_SMR.1, FDP_ACC.2/RV_Heap, FMT_MSA.1/RV_Heap |

| | | |
|---|---|---|
| | (FMT_MSA.1) and (FMT_SMR.1) | |
| FMT_MSA.3/RV_Heap | (FMT_MSA.1) and (FMT_SMR.1) | FMT_SMR.1, FMT_MSA.1/RV_Heap |
| FMT_SMF.1/RV_Heap | No Dependencies | |
| FDP_ACC.2/RV_Transient | (FDP_ACF.1) | FDP_ACF.1/RV_Transient |
| FDP_ACF.1/RV_Transient | (FDP_ACC.1) and (FMT_MSA.3) | FDP_ACC.2/RV_Transient, FMT_MSA.3/RV_Transient |
| FMT_MSA.1/RV_Transient | (FDP_ACC.1 or FDP_IFC.1) and (FMT_SMF.1) and (FMT_SMR.1) | FMT_SMR.1, FDP_ACC.2/RV_Transient, FMT_SMF.1/RV_Transient |
| FMT_MSA.2/RV_Transient | (FDP_ACC.1 or FDP_IFC.1) and (FMT_MSA.1) and (FMT_SMR.1) | FMT_SMR.1, FDP_ACC.2/RV_Transient, FMT_MSA.1/RV_Transient |
| FMT_MSA.3/RV_Transient | (FMT_MSA.1) and (FMT_SMR.1) | FMT_SMR.1, FMT_MSA.1/RV_Transient |
| FMT_SMF.1/RV_Transient | No Dependencies | |

**Table 13 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 Java Card RE.

**The dependency FAU_SAA.1 of FAU_ARP.1 is discarded.** The dependency of FAU_ARP.1 on FAU_SAA.1 assumes that a "potential security violation" generates an audit event. On the contrary, the events listed in FAU_ARP.1 are self-contained (arithmetic exception, ill-formed bytecodes, access failure) and ask for a straightforward reaction of the TSFs on their occurrence at runtime. The JCVM or other components of the TOE detect

these events during their usual working order. Thus, there is no mandatory audit recording in this ST.

**The dependency FAU_GEN.1 of FAU_STG.2 is discarded.** The FAU_STG.2 is related to the patch. When the identification of the patch is incorrect, the TOE rise a kill Card exception. The FAU_GEN is then discarded as the card returns only the ATR. There is need to store any audit function.

**The dependency FIA_UAU.1 of FIA_AFL.1/PIN is discarded.** The TOE implements the firewall access control SFP, based on which access to the object implementing FIA_AFL.1/PIN is organized.

**The dependency FIA_UAU.1 of FIA_AFL.1/GP_PIN is discarded.** The TOE implements the firewall access control SFP, based on which access to the object implementing FIA_AFL.1/GP_PIN is organized.

#### 7.3.3.2 SARs Dependencies

| Requirements | CC Dependencies | Satisfied Dependencies |
|---|---|---|
| ADV_ARC.1 | (ADV_FSP.1) and (ADV_TDS.1) | ADV_FSP.5, ADV_TDS.5 |
| ADV_FSP.5 | (ADV_IMP.1) and (ADV_TDS.1) | ADV_IMP.2, ADV_TDS.5 |
| ADV_IMP.2 | (ADV_TDS.3) and (ALC_CMC.5) and (ALC_TAT.1) | ADV_TDS.5, ALC_CMC.5, ALC_TAT.3 |
| ADV_INT.3 | (ADV_IMP.1) and (ADV_TDS.3) and (ALC_TAT.1) | ADV_IMP.2, ADV_TDS.5, ALC_TAT.3 |
| ADV_TDS.5 | (ADV_FSP.5) | ADV_FSP.5 |
| AGD_OPE.1 | (ADV_FSP.1) | ADV_FSP.5 |
| AGD_PRE.1 | No Dependencies | |
| ALC_CMC.5 | (ALC_CMS.1) and (ALC_DVS.2) 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.3 | (ADV_IMP.1) | ADV_IMP.2 |
| ALC_FLR.1 | No Dependencies | |
| 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.3 | (ADV_FSP.2) and (ATE_FUN.1) | ADV_FSP.5, ATE_FUN.2 |
| ATE_DPT.3 | (ADV_ARC.1) and (ADV_TDS.4) and (ATE_FUN.1) | ADV_ARC.1, ADV_TDS.5, ATE_FUN.2 |
| ATE_FUN.2 | (ATE_COV.1) | ATE_COV.3 |
| 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.3, ATE_FUN.2 |
| 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.2, ADV_TDS.5, AGD_OPE.1, AGD_PRE.1, ATE_DPT.3 |

**Table 14 SARs Dependencies**

### *7.3.4 Rationale for the Security Assurance Requirements*

The ID-One Cosmo X product claims a conformance to the Common Criteria level EAL6, augmented with the component ALC_FLR.1.

### *7.3.5 ALC_FLR.1 Basic flaw remediation*

The Flaw remediation assurance improves a rigourous management for updating the TOE in the context of sensible market where the Javacard Plateform need a long life cycle to embed the additionnal applications.

# 8 TOE Summary Specification

## 8.1 TOE Summary Specification

### *8.1.1 TOE Security functions*

**SF_ATOMIC_TRANSACTION**

This TSF provides means to execute a sequence of modifications and allocations on the persistent memory so that either all of them are completed, or the TOE behaves as if none of them had been attempted. The transaction mechanism is used for updating internal TSF data as well as for performing different functions of the TOE, like installing a new package on the card. This TSF is also available for applet instances through the javacard.framework.JCSystem, javacard.framework.Util and javacardx.framework.util.ArrayLogic classes. The first class provides the applet instances with methods for starting, aborting and committing a sequence of modifications of the persistent memory. The other classes provide methods for atomically copying arrays. This TSF ensures that the following data is never updated conditionally:

- The validated flag of the PINs
- The validated flag of the BIO template
- The reason code of the CardException and CardRuntimeException
- Transient objects
- Global arrays, like the APDU buffer and the buffer that the applet instances use to store installation data
- Any intermediate result state in the implementation instance of the Checksum, Signature, Cipher, and Message Digest classes of the JavaCard API.

This TSF is in charge of setting back the state of the persistent memory as it was before they were started, when the following operations specified are not completed:

- Loading and linking of a package
- Installing a new applet instance
- Deleting a package
- Deleting an applet instance
- Collecting unreachable objects
- Reading from and writing to a static field, instance field or array position
- Populating, updating or clearing a cryptographic key
- Modifying a PIN value

Upon deallocation of a resource from any reference to an object instance created during an aborted transaction, any previous information content of the resource is made unavailable.

Finally, this TSF ensures that no transaction is in progress when a method of an applet instance is invoked for installing, deselecting, selecting or processing an APDU sent to the applet instance. Concerning memory limitations on the transaction journal, this TSF guarantees that an exception is thrown when the maximal capacity is reached. The TSF preserves a secure state when such limit is reached. Atomic Transactions are detailed in

the chapter Atomicity and Transactions of the **[JCRE]** and in the documentation associated to the JCSystem class in the **[JCAPI]**.

**SF_UNOBSERVABILITY**

This function assures that processing based on secure elements of the TOE does not reveal any information on those elements. For example, observation of a PIN verification cannot reveal the PIN value, observation a cryptographic computation cannot give information on the key.

**SF_SIGNATURE**

This TSF provides the applet instances with a mechanism for generating an electronic signature of a byte array content and verifying an electronic signature contained in a byte array. An electronic signature is made of a hash value of the information to be signed encrypted with a secret key. The verification of the electronic signature includes decrypting the hash value and checking that it actually corresponds to the block of signed bytes.

The signature algorithms are available to the applets through the javacard.Signature class of the Java Card API, ISOSecureMesssaging class and SecureChannel class. The length of the key to be used for the signature is defined by the applet instance when the key is created. Before generating the signature, the TSF verifies that the specified key is suitable for the operation (secret keys for signature generation), that it has been previously initialized, and that is in accordance with the specified signature algorithm (DES, RSA, etc). The TSF also checks that it has been provided with all the information necessary for the signature operation. For those algorithms that do not pad the messages, the TSF checks that the information to be signed is block aligned before performing the signature operation. Once the signature operation is performed, the internal TSF data used for the operation like the ICV is cleared. Signature operations are implemented to resist to environmental stress and glitches and include measures for preventing information leakage through covert channels.

Mechanisms of signature for Secure Messaging are available to the applets through the SecureChannel (Global Platform Card 2.2" specification) and ISOSecureMessaging (Proprietary API) classes. The signature is included in Data Objects.

**SF_SECURITY_FUNCTIONS_OF_THE_IC**

The TOE uses the security functions of the IC. The list of the security function is presented in the ST lite of the IC component.

**SF_RUNTIME_VERIFIER**

This security functionality ensures the secure processing of information by ensuring the following elements:

- Stack Control
- Heap Control
- Transient Control

Information on the processing is described on the related FDP_ACF.1.

**SF_RESIDENT_APPLICATION_DISPATCHER**

During prepersonalisation phase, this function tests for every command if manufacturer authentication is required.

**SF_FLEXICODE**

This TSF provides means to configure the TOE by deleting parts of code. These parts of code, called modules, are suppressed independently to preserve the TOE integrity. The TSF manages the list of modules embedded in the TOE, the rights to access to this functionnality and the rights to delete the modules.

**SF_RANDOM_NUMBER**

This TSF provides to card manager, resident application, applets a mechanism for generating challenges and key values. Random number generators are available to the applets through the RandomData class of the Java Card API. Off-card entity authentication is achieved through the process of initiating a Secure Channel and provides assurance to the card that it is communicating with an authenticated off-card entity. If any step in the off-card authentication process fails, the process shall be restarted (i.e. new session keys generated). The Secure Channel initiation and off-card entity authentication implies the creation of session keys derived from card static key(s).

**SF_PREPERSO_AND_PATCHING**

This function is in charge of pre-initializing the internal data structures, loading the configuration of the card, load or extracting the personalization block memories (Perso Dump) and loading patch code, if needed. The patch contains its identification elements that are used, during audit, to uniquely identify loaded code.

**SF_MEMORY_FAILURE**

When using the non volatile memory, in case of a bad writing, internal mechanisms are implemented to prevent an incoherency of the written data. In case of an impossible writing, an exception is raised.

**SF_MESSAGE_DIGEST**

This TSF provides the applet instances with a mechanism for generating an (almost) unique value for a byte array content. That value can be used as a short representative of the information contained in the whole byte array. The hashing algorithms are available to the applets through the MessageDigest class of the Java Card API. Before generating the hash value, the TSF verifies that it has been provided with all the information necessary for the hashing operation. For those algorithms that do not pad the messages, the TSF checks that the information is block aligned before computing its hash value.

**SF_MANUFACTURER_AUTHENTICATION**

At prepersonalisation phase, manufacturer authentication at the beginning of a communication session is mandatory prior to any relevant data being transferred to the TOE. The manufacturer uses the random number returned in the INITIALIZE AUTHENTICATION PROCESS Command APDU. EXTERNAL AUTHENTICATE Command APDU is used to verify the cryptogram computed from the challenge by the manufacturer.

The max number of unsuccessful authentication attempts (basically 3) is described on the related FIA_AFL.1/PP.

**SF_KEY_MANAGEMENT**

This function enables key sets management (PIN, BIO). It allows creating updating and deleting key sets. It is used to load keys to the card. It also implements verification of Key sets attributes: key lengths, key types... and enforces the loaded keys integrity

**SF_KEY_GENERATION**

This TSF enforces the creation and/or the oncard generation of all the cryptographic keys of the card using the method specified in that SFR.

**SF_KEY_DESTRUCTION**

This TSF disables the use of a key both logically and physically. When a key is cleared, the internal life cycle of the key container is moved to a state in which no operation is allowed. Applet instances may invoke this TSF through the interfaces declared in the javacard.security package of the Java Card API.

**SF_KEY_AGREEMENT**

This TSF provides the applet instances with a mechanism for supporting key agreement algorithms such as EC Diffie-Hellman [IEEE P1363].

**SF_JBOX**

The SF_JBOX provides an environment to securely execute native code from third parties. SF_JBOX ensures that only program code and data contained in the JBox can be accessed from within this JBox and therefore cannot harm, manipulate, or influence other parts of the TOE. This fulfills the SFRs FDP_ACC.2/ JBox, FDP_ACF.1/JBox and FMT_MSA.1/JBox. Access to the CPU mode, memory outside the JBox, the MMU segment table, and dedicated registers which allow configuration of the MMU and allow system management is prohibited for code executed in the JBox to fulfill FDP_ACF.1/JBox. The MMU segment table to configure the MMU is part of the JBox which fulfils FMT_MSA.3/JBox This MMU segment table can be modified during the prepersonalization in accordance with FMT_MSA.3/JBox to specify alternative settings for initially restrictive values for the MMU segment table. This supports FMT_SMF.1/JBox.

**SF_HARDWARE_OPERATING**

When needed, at each start up or before first use, a self test of each hardware functional module is done, i.e.: DES, RSA, RNG implements a know calculus and checks if the result is correct. When executing, external hardware event can be trigged to prevent attacks or bad use. Temperature, frequency, voltage, light, glitch are considered as abnormal environmental conditions and put the card in frozen state. The TOE shall monitor IC detectors (e.g. out-of-range voltage, temperature, frequency, active shield, memory aging) and shall provide automatic answers to potential security violations through interruption routines that leave the device in a secure state.

The TOE with the IC has detectors of operational conditions. It shall resist to attackers with high-attack potential according to **[JIL1]** characterisation, in particular, to leakage attacks,

intrusive (e.g. probing, fault injection) and non-intrusive (e.g. SPA, DPA, EMA) attacks, operational conditions manipulation (voltage, clock, temperature, etc) and physical attacks aiming at modification of the IC content or behaviour. To be compliant to related SUN Protection Profile **[PP0099]**, the off-card verifier is mandatory in this ST; however, this TOE runs some additional verification at execution time. These verifications ensure that: 1. No read accesses are made to Java Card System code, data belonging to another application, data belonging to the Java Card System, 2. No write accesses are made to another application's code, Java Card System code, another application's data Java Card System or API data, 3. No execution of code is done from a method or from a method fragment belonging to another package (including execution on arbitrary data).

**SF_GP_DISPATCHER**

While a Security Domain is selected, this function tests for every command, according to the Security Domain life cycle state and the Card life cycle state, if security requirements are needed (if a Secure Channel is required).

**SF_FIREWALL**

This TSF enforces the Firewall security policy and the information flow control policy at runtime. The former policy controls object sharing between different applet instances, and between applet instances and the Java Card RE. The latter policy controls the access to global data containers shared by all applet instances. This TSF is enforced by the Java Card platform Virtual Machine (Java Card VM). During the execution of an applet, the Java Card VM keeps track of the applet instance that is currently performing an action. This information is known as the currently active context. Two kinds of contexts are considered: applet instances contexts and the Java Card RE context, which has special privileges for accessing objects. The TSF makes no difference between instances of applets defined in the same package: all of them belong to the same active context. On the contrary, instances of applets defined in different packages belong to different contexts. Each object belongs to the context that was active when the object was allocated. Initially, when the Java Card VM is launched, the context corresponding to the applet instance selected for execution becomes the first active context. Each time an instance method is invoked on an object, a context switch is performed, and the owner of the object becomes the new active context. On the contrary, the invocation of a static method does not entail a context switch. Before executing a bytecode that accesses an object, the object's owner is checked against the currently active context in order to determine if access is allowed. Access is determined by the Firewall access control rules specified in the chapter Applet Isolation and Object Sharing of the **[JCRE]**. Those rules enable controlled sharing of objects through interface methods that the object's owner explicitly exports to other applet instances, and provided that the object's owner explicitly accepts to share it upon request of the method's invoker.

**SF_EXCEPTION**

In case of abnormal event: data unavailable on an allocation, illegal access to a data, the system owns an internal mechanism that allows to stop the code execution and raise an exception.

**SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL**

Off-card entity authentication is achieved by initiating a Secure Channel and provides assurance to the card that it is communicating with an authenticated off-card entity. If any step in the off-card authentication process fails, the process shall be restarted (i.e. new session keys generated). The Secure Channel initiation and off-card entity authentication implies the creation of session keys derived from card static key(s).

**SF_ENCRYPTION_AND_DECRYPTION**

This TSF provides the applet instances with mechanisms for encrypting and decrypting the contents of a byte array.

The ciphering algorithms are available to the applets through the Cipher class of the Java Card API, ISOSecureMessaging class and SecureChannel class. The length of the key to be used for the ciphering operation is defined by the applet instance when the key is generated. Before encrypting or decrypting the byte array, the TSF verifies that the specified key has been previously initialized, and that is in accordance with the specified ciphering algorithm (DES, RSA, etc). The TSF also checks that it has been provided with all the information necessary for the encryption/decryption operation. Once the ciphering operation is performed, the internal TSF data used for the operation like the ICV is cleared. Ciphering operations are implemented to resist to environmental stress and glitches and include measures for preventing information leakage through covert channels.

Mechanisms of encrypting and decrypting for Secure Messaging are available to the applets through the SecureChannel (Global Platform Card 2.2" specification) and ISOSecureMessaging (Proprietary API **[AGD_PAPI]**) classes.

**SF_DATA_INTEGRITY**

Some of the data in non volatile memory can be protected. Keys, PIN, BIO templates package and patch code are protected with integrity value. When reading and writing operation, the integrity value is checked and maintained valid. In case of incoherency, an exception is raised to prevent the bad use of the data. SecureStore is a mean for protecting JavaCard data in integrity.

**SF_DATA_COHERENCY**

As coherency of data should be maintained, and as power is provided by the CAD and might be stopped at all moment (by tearing or attacks), a transaction mechanism is provided. When updating data, before writing the new ones, the old ones are saved in a specific memory area. If a failure appears, at the next start-up, if old data are valid in the transaction area, the system restores them for staying in a coherent state.

**SF_DAP_VERIFICATION**

An Application Provider may require that its Application code to be loaded on the card is checked for integrity and authenticity. The DAP Verification privilege of the Application Provider's Security Domain detailed in Section 9.2.1 of provides this service on behalf of an Application Provider. A Controlling Authority may require that all Application code to be loaded onto the card shall be checked for integrity and authenticity. The Mandated DAP Verification privilege of the Controlling Authority's Security Domain detailed in Section 9.2.1 of provides this service on behalf of the Controlling Authority. The keys and algorithms to

be used for DAP Verification or Mandated DAP Verification are implicitly known by the corresponding Security Domain.

**SF_CLEARING_OF_SENSITIVE_INFORMATION**

This TSF clears all the data containers that hold sensitive information when that information is no longer used or upon the allocation of the resource. This includes:

- The contents of the memory blocks allocated for storing class instances, arrays, static field images and local variables, before allocating a fresh block
- The objects reclaimed by the Java Card VM garbage collector
- The code of the deleted packages
- The objects accessible from a deleted applet instance
- The content of the bArray argument of the Applet.install method after a new applet instance is installed
- The content of CLEAR ON DESELECT transient objects owned by an applet instance that has been deselected when no other applets from the same package are active on the card
- The content of all transient objects after a card reset
- The contents of the cryptographic buffer after performing cryptographic operations
- The Reference to an object instance created during an aborted transaction
- The validated flag of the PINs after a card reset
- The validated flag of the BIO templates after a card reset

*Application Note:*

This function is in charge of clearing the information contained in the objects that are no longer accessible from the installed packages and applet instances. Clearing is performed on demand of an applet instance through the JCSystem.requestObjectDeletion() method.

**SF_CARDHOLDER_VERIFICATION**

This TSF enables applet instances to authenticate the sender of a request as the true cardholder. Applet instances have access to these services through the OwnerPIN class. Cardholder authentication is performed using the following security attributes:

- A secret enabling to authenticate the cardholder
- The maximum number of consecutive unsuccessful comparison attempts that are admitted
- A counter of the number of consecutive unsuccessful comparison attempts that have been performed so far
- The current life cycle state of the secret (reference value). This state is always updated, even if the modification is in the scope of an open transaction. Each time an attempt is made to compare a value to the reference value, and prior to the comparison being actually performed, if the reference is blocked, then the comparison fails and the reference value is not accessed. Otherwise, the try counter is decremented by one. This operation is always performed, even if it is in the scope of an open transaction. If the comparison is successful, then the try counter is reset to the try limit. When the try counter reaches zero, the reference enters into a blocked state, and cannot be used until it is unblocked. Cardholder Verification Method services are implemented to resist to environmental stress and glitches and

include measures for preventing information leakage through covert channels. In particular, unsuccessful authentication attempts consume the same power and execution time than successful ones. The Cardmanager uses the class OwnerPin to provide the services to the Applet that want benefit of the Shared GP_PIN. The **SF_CARDHOLDER_VERIFICATION implements all Pin verifications: D.PIN, GP.PIN**.

**SF_CARD_MANAGEMENT_ENVIRONMENT**

This TSF is in charge of initializing and managing the internal data structures of the Card Manager. During the initialization phase of the card, this TSF creates the Installer and the Applet Deletion Manager and initializes their internal data structures. The internal data structures of the Card Manager includes the Package and Applet Registries, which respectively contains the currently loaded packages and the currently installed applet instances, together with their associated AIDs. This TSF is also in charge of dispatching the APDU commands to the applets instances installed on the card and keeping traces of which are the currently active ones. It therefore handles sensitive TSF data of other security functions, like the Firewall.

**SF_CARD_CONTENT_MANAGEMENT**

This TSF ensures the following functionalities:

- Loading (Section 9.3.5 of **[GP2]**): This function allows the addition of code to mutable persistent memory in the card. During card content loading, this TSF checks that the required packages are already installed on the card. If one of the required packages does not exist, or if the version installed on the card is not binary compatible with the version required, then the loading of the package is rejected. Loading is also rejected if the version of the CAP format of the package is newer than the one supported by the TOE. If any of those checks fails, a suitable error message is returned to the CAD.
- Installation (Section 9.3.6 of **[GP2]**): This function allows the Installer to create an instance of a previously loaded Applet subclass and make it selectable. In order to do this, the install() method of the Applet subclass is invoked using the context of that new instance as the currently active context. If this method returns with an exception, the exception is trapped and the smart card rolls back to the state before starting the installation procedure.
- Deletion (Section 9.5 of **[GP2]**): This function allows the Applet Deletion Manager to remove the code of a package from the card, or to definitely deactivate an applet instance, so that it becomes no longer selectable. This TSF performs physical removal of those packages and applet data stored in NVRAM, while only logical removal is performed for packages in ROM. This TSF checks that the package or applet actually exists, and that no other package or applet depends on it for its execution. In this case, the entry of the package or applet is removed from the registry, and all the objects on which they depend are garbage collected. Otherwise, a suitable error is returned to the CAD. The deletion of the Applet Deletion Manager, the Installer or any of the packages required for implementing the Java Card platform Application Programming Interface (Java Card API) is not allowed.
- Extradition (Section 9.4.1 of **[GP2]**): This function allows the Installer to associate load files or applet instances to a Security Domain different than their currently associated Security Domain. It is also used to associate a Security Domain to another

Security Domain or to itself thus creating Security Domains hierarchies. If this method returns with an exception, the exception is trapped and the smart card rolls back to the state before starting the extradition procedure.

- Registry update (Section 9.4.2 of **[GP2]**): This function allows the Installer to populate, modify or delete elements of the Registry entry of applet instances. If this method returns with an exception, the exception is trapped and the smart card rolls back to the state before starting the extradition procedure.

### *8.1.2 Security functions for PACE*

**SF.PACE - Authentication using PACE**

This TSF provides the Password Authenticated Connection Establishment Authentication (all mappings) as described in [ICAO_9303] and session keys generation to be used with the support of SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL, as described in [ICAO_9303]. SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL is in charge of Secure Messaging used with PACE. In case the number of consecutive failed authentication attempts crosses the administrator defined number defined (used in FIA_AFL.1/PACE) the TSF will slow down further authentication attempts. The PACE can be initialized on Javacard Logical Channels. A PACE session is then available on all logical channels: the global policy for PACE (GPP) is established and the platform context is associated to PACE. Communication with the platform is then performed by secure messaging (C-MAC | C-ENC | R-MAC | R-ENC). The platform automatically unwraps and wraps APDUs with SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL. The applet manages its I/O in clear using the regular APDU class. GPP could be paused on one logical channel by an applet that needs to switch to another Privacy Protocol.

The PACE can be used by an applet thru only one Logical Channel, the Applet selection is then protected by the PACE privacy protocol (LPP). The applet is declared as a trusted applet. The PACE is integrated to the application context. Automatic Secure Messaging (like the one in GPP) is available thru a proprietary constant. The applet selection is supported by SF.FIREWALL. The applet is selected if the PACE with associated password has been performed.

All SFRs from PACE package are related to this SF for data protection, authentication and secure messaging.

### *8.1.3 Additional Security Function for MODULES*

The following specification is dedicated to modules managed under FLEXICODE IDEMIA mechanism.

**SF.PACE_CAM - Authentication using PACE_CAM**

This security functionality ensures the PACE_CAM is correctly performed. It can only be performed once the TOE is personalized with:

- the chip authentication mapping (CAM) keys the Personalization Agent loaded during the personalization phase
- the PACE password. Furthermore, this security functionality ensures the correct calculation of the PACE CAM session keys.

This security function is linked to SF.PACE

## 8.2 SFRs and TSS

### *8.2.1 SFRs and TSS - Rationale*

**CoreG_LC Security Functional Requirements**

*Application Programming Interface*

**FCS_CKM.1** This requirement is fulfilled by SF_KEY_GENERATION. It enforces the creation and/or the oncard generation of all the cryptographic keys of the card.

**FCS_CKM.4** SF_KEY_DESTRUCTION fulfils this SFR, it enforces the destruction of all the cryptographic keys of the card using the method specified in that SFR.

**FCS_COP.1** This SFR is verified by the following set of Security functionalities:

- All signature and verification operation by RSA, TDES and AES are fulfilled by SF_SIGNATURE, also fulfilled by SF_KEY_AGREEMENT by providing the applet instances with a mechanism for supporting key agreement algorithms such EC Diffie-Hellman **[IEEE]**.
- This requirement by using SF_ENCRYPTION_AND_DECRYPTION provides the applet instances with a mechanism for encrypting and decrypting the contents of a byte array.
- SF_SIGNATURE permits to hash functions with SHA-1, SHA-224, SHA-256, SHA-384 (when SHA3 is embedded) and SHA-512. It is also fulfilled by SF_MESSAGE_DIGEST by providing applet instances with a mechanism for generating an (almost) unique value for the contents of a byte array. Also fulfilled by SF_KEY_AGREEMENT by providing the applet instances with a mechanism for supporting key agreement algorithms such as EC Diffie-Hellman **[IEEE]**.

**FDP_RIP.1/ABORT** Any reference to an object instance created during an aborted transaction- see SF_ATOMIC_TRANSACTIONS- is cleaned by using SF_CLEARING_OF_SENSITIVE_INFORMATION.

**FDP_RIP.1/APDU** The TSF SF_CLEARING_OF_SENSITIVE_INFORMATION enforces the clearing of the previous contents of the APDU buffer before processing a new APDU.

**FDP_RIP.1/GlobalArray** The TSF SF_CLEARING_OF_SENSITIVE_INFORMATION enforces the clearing of the previous contents of the buffer. The array is no longer available to any

applet and is deleted and the memory in use by the array is cleared and reclaimed in the next object deletion cycle.

**FDP_RIP.1/bArray** The TSF SF_CLEARING_OF_SENSITIVE_INFORMATION enforces the clearing of the previous contents of the buffer containing the installation data of an applet instance before installing a new one.

**FDP_RIP.1/KEYS** In order to perform a cryptographic operation, the key involved in the operation has to be copied out of its secure container into the cryptographic buffer of the IC co-processor. This function is in charge of ensuring that such buffer is cleared immediately after completing the operation, the clearing is done by SF_CLEARING_OF_SENSITIVE_INFORMATION.

**FDP_RIP.1/TRANSIENT** This function is in charge of clearing the information contained in the transient objects when a clearing event arrives (deselection or card reset), invoked by SF_CLEARING_OF_SENSITIVE_INFORMATION.

**FDP_ROL.1/FIREWALL** SF_ATOMIC_TRANSACTION, when the operations specified are not completed, this TSF is in charge of setting back the state of the persistent memory as it was before they were started. As required in chapter 7 of the **[JCRE]** and the **[JCAPI]**, this TSF does not undo those modifications performed on the RAM, like the modification of the APDU buffer, the installation buffer, the transient objects, the try counters of the PINs and the reason code of the card exceptions. If the commit capacity is reached, this TSF prevents any further modification of the persistent memory.

**FCS_RNG.1** SF_RANDOM_NUMBER: This TSF is in charge of providing random numbers. SF_SECURITY_FUNCTIONS_OF_THE_IC: The seed come from the IC random generation.

*Firewall Policy*

**FDP_ACC.2/FIREWALL** The access control policy is ensured by SF_FIREWALL, it controls whether an instance of an applet class declared in a package (subject) may read, write or execute an instance method (operations) of an object (object).

**FDP_ACF.1/FIREWALL** FIREWALL Security attribute based access control -which security attributes is attached to which subject/object of the policy- is specified in the SF_FIREWALL.

**FDP_IFC.1/JCVM** This requirement is fulfilled by SF_FIREWALL, this TSF enforces the information flow control rules of Firewall security policy. It controls whether an applet

instance or javacard RE (subject) may store into persistent memory a reference of a global shared data container (objects).

**FDP_IFF.1/JCVM** This requirement is fulfilled by SF_FIREWALL. This TSF controls operations, based on current active context implemented in SF_FIREWALL.

**FDP_RIP.1/OBJECTS** SF_CLEARING_OF_SENSITIVE_INFORMATION. The TSF clears the contents of the freshly allocated objects before releasing the object to the applet. On the TSF, memory is cleared when the object is removed during Garbage Collection. All this TSFI lead to Garbage Collection

**FMT_MSA.1/JCRE** SF_FIREWALL When an instance method is applied to an object, this TSF is in charge of performing a context switch to the context of the object's owner. The TSF is also in charge of dispatching the APDU commands to the applets instances installed on the card and keeping trace of which are the currently active ones.

**FMT_MSA.1/JCVM** SF_FIREWALL When an instance method is applied to an object, this TSF is in charge of performing a context switch to the context of the object's owner. The TSF is also in charge of dispatching the APDU commands to the applets instances installed on the card and keeping traces of which are the currently active ones.

**FMT_MSA.2/FIREWALL_JCVM** SF_FIREWALL When an applet instance is selected for execution, this TSF initializes the currently active context with (the context of) that instance. Applet selection includes the verification that the instance actually exists on the card. Then, during the execution of the Java Card VM, this TSF propagates that secure value the other security attributes involved in the Firewall policy (object's owner).

**FMT_MSA.3/FIREWALL** SF_FIREWALL The TSF initializes the security attributes of the Firewall and Java Card VM security policies when an applet instance is selected for execution, when an instance method is invoked and when an object is allocated. This TSF does not provide means for a subject to override those initial values.

**FMT_MSA.3/JCVM** SF_FIREWALL. The TSF initializes the security attributes of the Firewall and Java Card VM security policies when an applet instance is selected for execution, when

an instance method is invoked and when an object is allocated. This TSF does not provide means for a subject to override those initial values.

**FMT_SMF.1** This SFR is fulfilled by SF_CARD_CONTENT_MANAGEMENT, when an instance method is applied to an object; this TSF is in charge of performing a context switch to the context of the object's owner.

**FMT_SMR.1** This requirement is full filled by SF_FIREWALL, this TSF uses a special value for the currently active context that identifies the Java Card RE (JCRE) and Java Card VM (JCVM).

*Card Security Management*

**FAU_ARP.1** This SFR is enforced by the following TSF:

- The SF_FIREWALL throws an instance of the SecurityException class when an attempt to violate a security policy rule is detected.
- The SF_EXCEPTION ensures that all cases of exceptions are thrown and managed during javacard runtime environment execution.

**FDP_SDI.2/DATA** The TSF SF_DATA_INTEGRITY ensures integrity of PIN, Keys and application code (package)(CRC 16). A loss of integrity increases killcard counter.

**FPR_UNO.1** The TSF SF_UNOBSERVABILITY ensures no user is able to observe PIN values when authentication of the cardholder.

**FPT_FLS.1** This SFR is enforced by the following TSF:

- SF_ATOMIC_TRANSACTIONS: card tearing and power failures and abortion of a transaction in an unexpected context
- SF_FIREWALL: violations of the Firewall access control rules,
- SF_CARD_CONTENT_MANAGEMENT: insufficient resources to install a package and CAP file inconsistency errors.
- SF_CLEARING_OF_SENSITIVE_INFORMATION: ensures the erase of previous information stored, like the flags of pin or reason code contained in the CardException or CardRuntimeException.

**FPT_TDC.1** This SFR is fulfilled by F_CARD_MANAGEMENT_ENVIRONMENT. It interprets cap files: bytes code and data arguments.

*AID Management*

**FIA_ATD.1/AID** This SFR is fulfilled by SF_ARD_CONTENT_MANAGEMENT: It controls the addition of new entries in the Applet Registry. Each time a new entry is added, the TSF

controls that it contains the information specified in that SFR. This is done on CAP FILE LOADING and applet installation.

**FIA_UID.2/AID** The TSF SF_FIREWALL identifies the applet instance requesting access to objects through the currently active context. Retrieving the currently active context always precedes the execution of the bytecodes under the control of the Firewall, as this information is required for checking the premises of its access control rules.

**FIA_USB.1/AID** The TSF SF_FIREWALL uses the security attribute introduced in the SFR to check whether an applet instance (subject) representing an Application Provider (user) may access an object through the firewall.

**FMT_MTD.1/JCRE** SF_CARD_CONTENT_MANAGEMENT fulfils this SFR, it controls the creation of new applet instances on the card. Each time an applet instance is created, the Installer adds an entry for it in the Applet Registry

**FMT_MTD.3/JCRE** This SFR is fulfilled by SF_CARD_CONTENT_MANAGEMENT: it controls that only secure values are assigned as attributes of an applet instance. Invalid AIDs for the applet instances, like an AID that is already in use, are also rejected

**InstG Security Functional Requirements**

**FDP_ITC.2/Installer** This SFR is implemented by SF_CARD_CONTENT_MANAGEMENT: The SF ensures safe CAP FILE LOADING and applet installation process. It modifies the CAP files to produce the TOE intern representation of the loaded package. It also performs coherency checks on the CAP files and verifies the export references.

**FMT_SMR.1/Installer** This SFR is implemented by SF_CARD_CONTENT_MANAGEMENT: The TSF is in charge of creating the applet instance that plays the role of the Applet Installation Manager.

**FPT_FLS.1/Installer** This SFR is fulfilled by the following SF:

- The SF_CARD_CONTENT_MANAGEMENT: is in charge of checking that all the conditions for safely installing a package or an applet instance are fulfilled during the installation procedure. If conditions cannot be verified the installation is deemed unsuccessful and either an exception is thrown or the card is frozen, depending of the failure severity. Card tearing or reset also cause an installation failure.
- SF_ATOMIC_TRANSACTIONS is in charge of rolling back to a secure state when the installation of a package or an applet instance is aborted
- This function is in charge of clearing the information contained in the packages that is not necessary for the execution of the code of the applet invoked by SF_CLEARING_OF_SENSITIVE_INFORMATION.

**FPT_RCV.3/Installer** This SFR is fulfilled by the following SF:

- SF_CARD_CONTENT_MANAGEMENT: In case of severe failure during package or applet installation, the card is frozen (KillCard). Such failures (for example the

loading of a CAP file with an invalid format) are considered as security problems. The maintenance mode is represented by the frozen state of the card. The secure state is then reached on next card reset where Garbage Collector is launch to retrieve lost memory and where the transaction mechanism allows retrieving the initial state.

- SF_ATOMIC_TRANSACTION: The TSF is in charge of rolling back to a secure state when the installation of a package or an applet instance is aborted.

**ADELG Security Functional Requirements**

**FDP_ACC.2/ADEL** The access control policy for deletion is made by SF_CARD_CONTENT_MANAGEMENT, it controls whether the Applet Deletion Manager (subject) may delete (operation) a package or an applet instance (object).

**FDP_ACF.1/ADEL** The access control policy for deletion is made by SF_CARD_CONTENT_MANAGEMENT, it controls whether the Applet Deletion Manager (subject) may delete (operation) a package or an applet instance (object).

**FDP_RIP.1/ADEL** The TSF SF_CLEARING_OF_SENSITIVE_INFORMATION renders inaccessible the code of a deleted package and the class instances and arrays allocated by a deleted applet instance.

**FMT_MSA.1/ADEL** The ADEL access policy is implemented in SF_CARD_CONTENT_MANAGEMENT, this TSF keeps track of which applet instances are currently active on which logical channels. Only the Card Manager (which in **[PP0099]** is identified with the Java Card RE role) is allowed to associate or remove the association between an applet instance and a logical channel. These actions are performed as part of command dispatching

**FMT_MSA.3/ADEL** The SF_CARD_CONTENT_MANAGEMENT enforces the assignment of restrictive values for the security attributes of the Applet Deletion policy.

**FMT_SMF.1/ADEL** Modifying the active applet security context is done by SF_CARD_MANAGEMENT_ENVIRONMENT, it's allowed to card manager.

**FMT_SMR.1/ADEL** This SFR is fulfilled by SF_CARD_MANAGEMENT_ENVIRONMENT: it keeps track of which applet instances are currently active on which logical channels. Only the Card Manager is allowed to associate or remove the association between an applet instance and a logical channel.

**FPT_FLS.1/ADEL** This SFR is ensured by the following TSF:

- SF_CARD_CONTENT_MANAGEMENT is in charge of checking that all the conditions for safely deleting a package or an applet instance are fulfilled before starting the deletion procedure.
- SF_ATOMIC_TRANSACTION: This TSF is in charge of rolling back to a secure state when the deletion of a package or an applet instance is aborted.

- SF_CLEARING_OF_SENSITIVE_INFORMATION: is in charge of checking that all the conditions for safely deleting a package or an applet instance are fulfilled before starting the deletion procedure.

**ODELG Security Functional Requirements**

**FDP_RIP.1/ODEL** This SFR is met by SF_CLEARING_OF_SENSITIVE_INFORMATION: This TSF renders inaccessible the code of a deleted package and the class instances and arrays allocated by a deleted applet instance.

**FPT_FLS.1/ODEL** The TSF SF_CLEARING_OF_SENSITIVE_INFORMATION is in charge of checking that all the conditions for safely deleting a package or an applet instance are fulfilled before starting the deletion procedure.

**CarG Security Functional Requirements**

*Miscellaneous*

**FCO_NRO.2/CM** During the loading phase, the SF_CARD_CONTENT_MANAGEMENT: controls card content loading, it verifies the proof of the origin of the Load File. Before to start the loading, the open checks that the user is authenticated, checks the presence of the < DAPBlock > in the < LoadFile >, requires the Security Domain Verifier to verify it.

**FDP_IFC.2/CM** The rule of the CAP FILE LOADING flow control policy is specified by SF_CARD_CONTENT_MANAGEMENT: it verifies that all the loading commands are issued in the Secure Channel session. It compares the Load File Data Block Hash present in the command install for load against the received. It also requires the Dap verification of all entities committed in the loading phase, ensured by SF_DAP_VERIFICATION.

**FDP_IFF.1/CM** This SFR is implemented by SF_DAP_VERIFICATION, it controls the communication protocol used by the CAD and the card for transmitting packages. The SFR is also implemented in the SF_CARD_CONTENT_MANAGEMENT to ensure the access control policy for the loading of the packages.

**FDP_UIT.1/CM** This SFR is implemented by SF_DAP_VERIFICATION, it controls imported data from modification, deletion, insertion, replay of some of the pieces of the application sent by the CAD. The verification is made by using: Encryption and decryption operations by SF_ENCRYPTION_AND_DECRYPTION function.

**FIA_UID.1/CM** The Security Functionality SF_GP_DISPATCHER met this SFR: While the Card manager (ISD) or Supplementary Security domain is selected, these functions test for every command if the secure channel is open. When the secure channel is not open then only these commands are available: Get data and Initialize Update. The initialize Update returns

to the user the key set version, Secure Channel identifier and the card random and the card cryptogram.

**FMT_MSA.1/CM** This SFR is implemented by two security functions: SF_KEY_MANAGEMENT: The TSF controls that only the CM can modify its key set and can change the card life cycle and set the default application SF_CARD_CONTENT_MANAGEMENT: This TSF controls whether the active entity has the privilege and the pre-authorization for make the Card Content Management operations, and that operation still available on the card. Its controls also that the card state allows the operations.

**FMT_MSA.3/CM** The TSF SF_CARD_CONTENT_MANAGEMENT provides the way to lock the Security Domain with Authorized Management privilege in order to restrict its card content management ability. This TSF provides also to disable permanently the Card Content Management operations for all entities on the card.

**FMT_SMF.1/CM** The TSF SF_CARD_CONTENT_MANAGEMENT controls whether the active entity has the privilege and the pre-authorization for making the Card Content Management operations - modify security attributes. -, and that operation still available on the card. Its controls also that the card state allows the operations.

**FMT_SMR.1/CM** The TSF SF_CARD_CONTENT_MANAGEMENT verifies that authentication is successful and the active entity has loading privilege (Authorized Management privilege) before processes any Card Content management command. The successful authentication proves the user identity and role.

**FTP_ITC.1/CM** Installing a new package is verified by SF_CARD_CONTENT_MANAGEMENT: the SF_GP_DISPATCHER tests if secure channel is required, and verification is made by SF_DAP_VERIFICATION.

*Additional Security Functional Requirements for CM*

**FPT_TST.1** This SFR is supported by the following TSF:

- SF_HARDWARE_OPERATING: At each start up, security function SF_Hardware_Operating is done. Random, DES, and CRC functional modules systematically tested: a known calculus is implemented and the result is checked. SHA, RSA, AES and ECC functional modules are tested at each start up or at first use, using the same method.
- SF_DATA_INTEGRITY: At each start up, the entire NVM integrity, so executable code, is checked. The NVM integrity is updated after patch loading so the next startup does not rise a kill card exception.
- SF_PACE: The TOE proposes to activate the PACE for all Logical Channels (GPP) and so at start-up the correct TOE operating is checked through SF_HARDWARE_OPERATING and SF_DATA_INTEGRITY

**FCO_NRO.2/CM_DAP** During the loading phase, SF_DAP_VERIFICATION verifies the proof of the origin of the Load File. Before to start the loading, the open checks that the user is authenticated, checks the presence of the < DAPBlock > in the < LoadFile >, requires the Security Domain Verifier to verify it.

**FIA_AFL.1/CM** This Requirement is fulfilled by SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL. It tests the result of authentication. By default the authentication result is assumed unsuccessful, so the authentication failure is recorded, the associated counter and the slowdown counter are incremented. If the authentication is successful, the authentication failure counter is decremented and the slowdown counter is reset.

**FIA_UAU.1/CM** This SFR is implemented by the following TSF:

- SF_GP_DISPATCHER: While the Card manager (ISD) or Supplementary Security domain is selected, these functions test for every command by SF_GP_DISPATCHER if the secure channel is open.
- SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL: When the secure channel is not open then only the command available are Get Data, Initialize Update, Select.

**FIA_UAU.4/CardIssuer** Present the use of CardIssuer authentication, function implemented in SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL, is given by using a RNG defined in SF_RANDOM_NUMBER.

**FIA_UAU.7/CardIssuer** This SFR is implemented by the following TSF SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL: It uses the key set version, Secure channel identifier and the card random and the card cryptogram for authentication. - SF_RANDOM_NUMBER: It permits CardIssuer Protected authentication feedback, no other information is given while the authentication is in progress.

**FPR_UNO.1/Key_CM** Import of keys are not observable by all subjects. This requirement is ensured by SF_KEY_MANAGEMENT and SF_UNOBSERVABILITY.

**FPT_TDC.1/CM** Key set and packages when imported are consistently interpreted by implementation of SF_KEY_MANAGEMENT.

**FMT_SMR.2/CM** The TSF SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL verifies that authentication is successful and the active entity has loading privilege before processes any Card Content management command. The successful authentication proves the user identity and role.

**FCS_COP.1/CM** The TSF SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL covers this SFR. It requires the cryptographic operations for the creation and management of secure

channel. The TSF SF_ENCRYPTION_AND_DECRYPTION provides a mechanism for encrypting and decrypting the contents of a byte array.

*Additional Security Functional Requirements for Resident application*

**FDP_ACC.2/PP** The SFR is implemented by the following TSF:

- SF_RESIDENT_APPLICATION_DISPATCHER: This TSF implements Access control policy for the resident application,
- SF_MANUFACTURER_AUTHENTICATION: This TSF is in charge of the card manufacturer authentication.

These TSF controls all access to all objects and all operations.

**FDP_ACF.1/PP** This SFR is met by the following TSF:

- SF_MANUFACTURER_AUTHENTICATION: The rules garanting access rights are made by SF_MANUFACTURER_AUTHENTICATION, it guarantees once the Prepersonalisation is authenticated, the card verifies for each action that the authentication is successful

**FDP_UCT.1/PP** This SFR is met by the following TSF:

- SF_MANUFACTURER_AUTHENTICATION: To transmit the InitKey (ISK) to Card Manager, user or subject must be successfully authenticated with the MSK. The transmitted InitKey is protected from disclosure by being ciphered by the MSK
- SF_PRE_PERSO_AND_PATCHING: This TSF is in charge of the prepersonalisation stage, this includes the patch loading.

**FDP_ITC.1/PP** The SFR is implemented by the following TSF:

- SF_MANUFACTURER_AUTHENTICATION enables trusted channel establishment thanks to authentication with the MSK and allows the prepersonalisation.
- SF_PRE_PERSO_AND_PATCHING enables to load patches, Perso dump and locks.
- SF_RESIDENT_APPLICATION_DISPATCHER During prepersonalisation phase, this function tests for every command if manufacturer authentication is required

**FIA_AFL.1/PP** The SFR is implemented by the following TSF:

- SF_MANUFACTURER_AUTHENTICATION: After 3 consecutive unsuccessful authentications a status error is always returned by the card.

**FIA_UAU.1/PP** The SFR is met by the following SF:

- SF_RESIDENT_APPLICATION_DISPATCHER: The set of command (INITIALIZE AUTHENTICATION PROCESS, GET DATA, MANAGE CHANNEL, SELECT APPLET) of the resident application can be performed without authentication.

**FIA_UID.1/PP** This SFR is implemented by the following TSF:

- SF_RESIDENT_APPLICATION_DISPATCHER: The set of command (INITIALIZE AUTHENTICATION PROCESS, GET DATA, MANAGE CHANNEL, SELECT APPLET) of the resident application can be performed without authentication.

**FMT_MSA.1/PP** This SFR is implemented by the following TSF:

- SF_MANUFACTURER_AUTHENTICATION: The MSK keys of the Card Manufacturer can be modified in Prepersonalisation phase after a successful authentication with the MSK.

**FMT_SMF.1/PP** This SFR is implemented by the following TSF:

- SF_MANUFACTURER_AUTHENTICATION: The MSK keys of the Card Manufacturer can be modified in Prepersonalisation phase after a successful authentication with the MSK.

**FIA_UAU.4/CardManu** This SFR is implemented by the following TSF:

- SF_MANUFACTURER_AUTHENTICATION: Prevents from Card Manufacturer authentication reuse during prepersonalisation

**FIA_UAU.7/CardManu** This SFR is implemented by the following SFR:

- SF_MANUFACTURER_AUTHENTICATION: During authentication, the command "INITIALIZE AUTHENTICATION PROCESS" is used to provide a random number implemented by the SF_RANDOM_NUMBER.
- Only Random number and NOK as result of authentication are provided to the user.

**FMT_MOF.1/PP** This SFR is implemented by the following TSF:

- SF_CARD_MANAGEMENT_ENVIRONMENT: Some commands and features of the resident application are active only during the Prepersonalisation Phase, such as the patch and lock loading. As soon as the Card Manager status is set to OP_READY this phase stops, and the commands and features are irreversibly disabled.
- SF_PRE_PERSO_AND_PATCHING: This function permits on Prepersonalisation phase to load patch code on the card.
- SF_RESIDENT_APPLICATION_DISPATCHER: During prepersonalisation phase, this function tests for every command if manufacturer authentication is required as commands used to load patches on the card.

**FMT_SMR.2/PP** This SFR is implemented by the following TSF:

- SF_PRE_PERSO_AND_PATCHING: This function permits on Prepersonalisation phase to load patch code on the card.
- SF_MANUFACTURER_AUTHENTICATION: After a successful authentication (of Card Manufacturer) using MSK, the TSF and card stay still in Prepersonalisation state.
- SF_RESIDENT_APPLICATION_DISPATCHER: During prepersonalisation phase, this function tests for every command according to the life cycle of the resident application

**FMT_MSA.3/PP** This SFR is implemented by the following TSF:

- SF_MANUFACTURER_AUTHENTICATION: This TSF implements Access control policy in preperso phase.
- SF_PRE_PERSO_AND_PATCHING: This TSF, once authenticated, implements prepersonalisations operations.

**FCS_COP.1/PP** At prepersonalisation phase, authentication cryptogram (signature computation and verification) are used by SF_MANUFACTURER_AUTHENTICATION. Data decryption (of patch, Dump Perso, locks or keys), integrity pattern verification (signature/MAC) are used by SF_PRE_PERSO_AND_PATCHING. Encrypted and decrypted data in bytes arrays are manipulated using SF_ENCRYPTION_AND_DECRYPTION. These functions call Cryptographic ones defined in previous FCS_COP operation SF_MANUFACTURER_AUTHENTICATION: This TSF implements Access control policy in preperso phase; and after prepersonnalisation phase, SF_GP_DISPATCHER implements Access control policy in user phase: A codop for package P can be loaded only by the SD owned package P.

**FCS_CKM.4/PP** This SFR is implemented by the following TSF:

- SF_KEY_DESTRUCTION: As soon as the Card Manager status is set to OP_READY, the MSK and LSK key is set to null (as the checksum is also to null) the key is not useful. The MSK after the first use is diversified, according to AGD_PRE **[AGD_PRE]**. The new version of the key replaces the previous one.

**FDP_UIT.1/PP** The SFR is implemented by the following TSF:

- SF_PRE_PERSO_AND_PATCHING: The data (patch, Dump Perso or locks) sent to the TOE are protected in integrity thanks to a signature computed by the TOE developer with the,dedicated key (JSK, LSK, MSK).

**FAU_STG.2** The SFR is implemented by the following TSF:

- SF_PRE_PERSO_AND_PATCHING: Upon request, the identification of the patch is returned.

*Additional Security Functional Requirements for SmartCard Platform*

**FPT_PHP.3/SCP** When executing, SF_HARDWARE_OPERATING shall resist changing operational conditions every times, external hardware event can be trigged to prevent attacks or bad use. Temperature, frequency, voltage, light, glitch are considered as abnormal environmental conditions and put the card in frozen state.

**FPT_RCV.4/SCP** The TSF SF_DATA_COHERENCY 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.

**FRU_FLT.1/SCP** When there is a failure of FLASH, in case of a bad writing, the SF_MEMORY_FAILURE implements internal mechanisms to prevent an incoherency of the written data. In case of an impossible writing, an exception is raised.

**FPR_UNO.1/USE_KEY** This SFR is implemented by the following TSF:

- SF_UNOBSERVABILITY: No user are able to observe keys whether the keys are in use.

*Additional Security Functional Requirements for the applets*

**FIA_AFL.1/PIN** This SFR is implemented by the following TSF:

- SF_CARDHOLDER_VERIFICATION: The TSF detects that the number of PIN presentations exceeds the maximum value previously configured. It blocks the PIN in this case. The entire OwnerPin class in involved in the process since it is used to set the PIN size and the maximum of successful tries, to verify the PIN, to reset the validation flag.

**FMT_MTD.2/GP_PIN** This SFR is implemented by the following TSF:

- SF_CARDHOLDER_VERIFICATION: The TOE ensures that the GlobalPin is blocked when its associated PIN try counter at reach the PIN try limit value.

**FMT_MTD.1/PIN** The SFR is implemented by the following TSF:

- SF_CARDHOLDER_VERIFICATION provides a complete PIN mechanism: change_default, query and modify to applets.

**FIA_AFL.1/GP_PIN** This SFR is implemented by the following TSF:

- SF_CARDHOLDER_VERIFICATION: The TOE checks that only the Card Manager and privileged application can change the pin try limit and Update the global Pin.

*Additional Security Functional Requirements for Runtime Verification*

Stack Control

**FDP_ACC.2/RV_Stack** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: The SF implements a complete access control on the Stack operations.

**FDP_ACF.1/RV_Stack** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: This SFR enforces the access conditions which guarantee the protection of the Stack.

**FMT_MSA.1/RV_Stack** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: This TSF implements the management of the security attributes.

**FMT_MSA.2/RV_Stack** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: This TSF ensures that only secure values for the attributes are accepted

**FMT_MSA.3/RV_Stack** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: This TSF implements the initialisation of the attributes of the access control policy.

**FMT_SMF.1/RV_Stack** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: The TSF specify the management function to modify the stack pointer. It controls the Stack and is able to change the associated parameter.

Heap Access

**FDP_ACC.2/RV_Heap** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: it implements the access control to the Heap.,

**FDP_ACF.1/RV_Heap** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: it ensures the access conditions to the Heap.

**FMT_MSA.1/RV_Heap** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: This TSF implements the management of the modification of the security attributes of the access control to the Heap.

**FMT_MSA.2/RV_Heap** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: This TSF implements the control that only security values are accepted for the security attributes..

**FMT_MSA.3/RV_Heap** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: This TSF implements initialisation of the security attributes.

**FMT_SMF.1/RV_Heap** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: The TSF directly controls the Heap and is able to change the associated parameter.

Transient Control

**FDP_ACC.2/RV_Transient** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: it implements the access control to guarantee the protection of Transient objects.

**FDP_ACF.1/RV_Transient** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: it implements access conditions and defines the security rules which guarantee the protection of Transient objects.

**FMT_MSA.1/RV_Transient** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: This TSF implements the management of the security attributes for the access control to the transient..

**FMT_MSA.2/RV_Transient** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: This TSF implements the condition that only secure attributes are accepted for the access control policy to the Transient.

**FMT_MSA.3/RV_Transient** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: This TSF controls that only restrictive values are accepted for security attributes used to enforce the transient access control policy.

**FMT_SMF.1/RV_Transient** This SFR is implemented by the following TSF:

- SF_RUNTIME_VERIFIER: The TSF directly controls the Transient and is able to change the associated parameter.

**JBox Security Requirements**

**FDP_ACC.2/JBox**

- SF_JBox: The TSF ensure the access control policy applied to the context that manages the execution environment of a piece of code (NVM, RAM, and resource accesses, etc.). It must not access: (i) the platform code, except through the authorized Native services (the available operations(5)), etc.); (ii) the platform data, except through shared memory

**FDP_ACF.1/JBox**

- SF_JBox: The TSF ensure the access control policy applied to the context that manages the execution environment of a piece of code (NVM, RAM, and resource accesses, etc.). The TSF controls the execution of native code in TPL context.

**FMT_MSA.1/JBox**

- SF_JBox: The TSF ensure the MPU/MMU is used and the configuration of MP/MMU can’t be modified.

**FMT_MSA.3/JBox**

- SF_JBox: The TSF ensure the MPU/MMU is used and the configuration of MP/MMU can’t be modified.

**FMT_SMF.1/JBox**

- SF_JBox: The TSF ensure a context switch to the context of the object's owner: the switch between TPL context and platform context allow changing MPU/MMU context to assigne RAM and NVM to the corresponding owner.

**Additional Security Requirements - FLEXICODE package**

*Additional Security Functionnal Requirements for API - DBI - Module*

**FMT_MTD.1/Activate_DBI** The SFR is implemented by the following TSF: SF_CARD_MANAGEMENT_ENVIRONMENT

This SFR is supported by SF_ATOMIC_TRANSACTION, SF_FIREWALL.

**FMT_MTD.1/Deactivate_DBI** The SFR is implemented by the following TSF: SF_CARD_MANAGEMENT_ENVIRONMENT

This SFR is supported by SF_FLEXICODE, SF_ATOMIC_TRANSACTION, SF_FIREWALL.

*Miscellaneous*

**FMT_MOF.1/FlxC** This SFR is implemented by the following TSF:

- SF_FLEXICODE: to delete the module with corresponding access rights and TOE integrity preservation.
- SF_RESIDENT_APPLICATION_DISPATCHER: This TSF implements Access control policy for the resident application,
- SF_MANUFACTURER_AUTHENTICATION: This TSF is in charge of the card manufacturer authentication.

**FMT_SMF.1/FlxC** This SFR is implemented by the following TSF:

- SF_FLEXICODE: to delete the module with corresponding access rights and TOE integrity preservation

**FDP_ACC.2/FlxC** The SFR is implemented by the following TSF:

- SF_FLEXICODE: to delete the module with corresponding access rights and TOE integrity preservation.
- SF_RESIDENT_APPLICATION_DISPATCHER: This TSF implements Access control policy for the resident application,
- SF_MANUFACTURER_AUTHENTICATION: This TSF is in charge of the card manufacturer authentication.

These TSF controls all access to all objects and all operations.

**FDP_ACF.1/FlxC** This SFR is met by the following TSF:

- SF_FLEXICODE: to delete the module with corresponding access rights and TOE integrity preservation.

- SF_MANUFACTURER_AUTHENTICATION: The rules garanting access rights are made by SF_MANUFACTURER_AUTHENTICATION, it guarantees once the Prepersonalisation is authenticated, the card verifies for each action that the authentication is successful

**FMT_MSA.1/FlxC** This SFR is implemented by the following TSF:

- SF_FLEXICODE: to delete the module with corresponding access rights and TOE integrity preservation.
- SF_MANUFACTURER_AUTHENTICATION: This TSF implements Access control policy in preperso phase.
- SF_PRE_PERSO_AND_PATCHING: This TSF, once authenticated, implements prepersonalisations operations.

**FMT_MSA.2/FlxC** This SFR is implemented by the following TSF:

- SF_FLEXICODE: to delete the module with corresponding access rights and TOE integrity preservation.
- SF_MANUFACTURER_AUTHENTICATION: This TSF implements Access control policy in preperso phase.
- SF_PRE_PERSO_AND_PATCHING: This TSF, once authenticated, implements prepersonalisations operations.

**FMT_MSA.3/FlxC** This SFR is implemented by the following TSF:

- SF_FLEXICODE: to delete the module with corresponding access rights and TOE integrity preservation.
- SF_MANUFACTURER_AUTHENTICATION: This TSF implements Access control policy in preperso phase.
- SF_PRE_PERSO_AND_PATCHING: This TSF, once authenticated, implements prepersonalisations operations.

**FMT_MSA.4/FlxC** This SFR is implemented by the following TSF:

- SF_FLEXICODE: to delete the module with corresponding access rights and TOE integrity preservation.
- SF_MANUFACTURER_AUTHENTICATION: This TSF implements Access control policy in preperso phase.
- SF_PRE_PERSO_AND_PATCHING: This TSF, once authenticated, implements prepersonalisations operations.

**FPT_FLS.1/FlxC** This SFR is implemented by the following TSF:

- SF_FLEXICODE: to delete the module with corresponding access rights and TOE integrity preservation.

**FDP_RIP.1/FlxC** This SFR is implemented by the following TSF:

- SF_FLEXICODE: to delete the module with corresponding access rights and TOE integrity preservation.

**Additional Security Requirements Monotonic Counters package**

**FDP_SDI.2/MONOTONIC_COUNTER** The TSF SF_DATA_INTEGRITY ensures integrity of monotonic counters. A loss of integrity increases killcard counter.

**Additional Security Functional Requirements for Sensitive Array package**

**FDP_SDI.2/ARRAY** The TSF SF_DATA_INTEGRITY ensures integrity of sensitive array. A loss of integrity increases killcard counter.

**Additional Security Functionnal Requirements for API - RSA - MODULE**

**FCS_CKM.1/RSA** This requirement is fulfilled by SF_KEY_GENERATION. It enforces the creation and/or the oncard generation of all the cryptographic keys of the card.

**FCS_COP.1/RSA** This SFR is verified by the following set of Security functionalities:

- All signatures and verification operations by RSA (when RSA module is embedded), are fulfilled by SF_SIGNATURE, also fulfilled by SF_KEY_AGREEMENT by providing the applet instances with a mechanism for supporting key agreement algorithms **[IEEE]**.
- This requirement by using SF_ENCRYPTION_AND_DECRYPTION provides the applet instances with a mechanism for encrypting and decrypting the contents of a byte array.
- SF_SIGNATURE permits to hash functions with SHA-1, SHA-224, SHA-256, SHA-384 (when SHA3 module is embedded) and SHA-512. It is also fulfilled by SF_MESSAGE_DIGEST by providing applet instances with a mechanism for generating an (almost) unique value for the contents of a byte array. Also fulfilled by SF_KEY_AGREEMENT by providing the applet instances with a mechanism for supporting key agreement algorithms such as EC Diffie-Hellman **[IEEE]**.

**Additional Security Functionnal Requirements for API - SHA3 - MODULE**

**FCS_COP.1/SHA3** This SFR is verified by the following set of Security functionalities:

- This requirement by using SF_ENCRYPTION_AND_DECRYPTION provides the applet instances with a mechanism for encrypting and decrypting the contents of a byte array.
- SF_SIGNATURE permits to hash functions with SHA-384 (when SHA3 module is embedded). It is also fulfilled by SF_MESSAGE_DIGEST by providing applet instances with a mechanism for generating an (almost) unique value for the contents of a byte array. Also fulfilled by SF_KEY_AGREEMENT by providing the applet instances with a mechanism for supporting key agreement algorithms such as EC Diffie-Hellman **[IEEE]**.

**FIA_AFL.1/BIO** This SFR is implemented by the following TSF:

- SF_CARDHOLDER_VERIFICATION: The TSF detects that the number of PIN_BIO presentations exceeds the maximum value previously configured. It blocks the PIN_BIO in this case.
- SF_DATA_INTEGRITY ensures the PIN_BIO and its template integrity.

**FMT_MTD.1/BIO** The SFR is implemented by the following TSF:

- SF_CARDHOLDER_VERIFICATION provides a complete PIN_BIO mechanism: change_default, query and modify to applets.

This SFR is supported by SF_ATOMIC_TRANSACTION,SF_DATA_INTEGRITY and SF_KEY_MANAGEMENT.

**Additional Security Functional Requirements for PACE package - PACE-DH - MODULE**

**FCS_CKM.1/DH_PACE_DH** The SFR is implemented by the following TSF: SF.PACE and SF.PACE_CAM implements the Diffie-Hellman algorithm needed for PACE protocole.

This SFR is supported by SF_FLEXICODE

**Additional Security Functionnal Requirements for HMAC - Module**

**FCS_COP.1/HMAC**

- This requirement by using SF_ENCRYPTION_AND_DECRYPTION provides the applet instances with a mechanism for encrypting and decrypting the contents of a byte array.
- SF_SIGNATURE permits to hash functions with SHA-256, SHA-384 (when SHA3 is embedded) and SHA-512 for generating an electronic signature. It is also fulfilled by SF_MESSAGE_DIGEST by providing applet instances with a mechanism for generating an (almost) unique value for the contents of a byte array.

#### 8.2.1.1 TOE Summary Specification

**Security functions for PACE**

**SF.PACE - Authentication using PACE** The TOE symmetric authentication functions is ensured by

- FIA_UID.1/PACE FIA_UAU.1/PACE, FIA_UAU.4/PACE, FIA_UAU.5/PACE, FIA_UAU.6/PACE for the PACE identification and authentication
- FIA_AFL.1/PACE for error management
- FMT_SMR.1/PACE for the role authentication

The Secure messaging is ensured by:

- FCS_CKM.1/DH_PACE, FCS_CKM.1/DH_PACE_DH, FTP_ITC.1/PACE FCS_RND.1/PACE for Securie messaging establishment

- FCS_COP.1/PACE_ENC, FCS_COP.1/PACE_MAC,FCS_CKM.4/DH_PACE and FDP_RIP.1/PACE. for data secure transfer through the Secure Messaging.

The TOE access control is ensured by

- for access control or none access to specific data FMT_MTD.1/INI_DIS, FMT_MTD.1/INI_ENA and FMT_MTD.1/KEY_READ,
- FMT_SMR.1/PACE for role management
- FMT_SMF.1/PACE for management functions linked to the states of the TOE

The protection of TOE data is ensured by

- FPT_EMS.1/PACE for the inherent leakage to TOE cryptographic operation
- FPT_TST.1 for the TOE test mechanisms as defined in
- FMT_LIM.1/PACE_Perso_Perso and FMT_LIM.2/PACE_Perso_Perso for protection against misuse of TOE test features
- FPT_PHP.3; FPT_FLS.1 for physical protection of the TOE and preservation of TOE secure state

**Additional Security Function for MODULES**

**SF.PACE_CAM - Authentication using PACE_CAM** The TOE authentication functions is ensured by

- FIA_AFL.1/PACE, Authentication failure handling PACE authentication using non-blocking authorisation data. The TOE increases the reaction time of the TOE after an unsuccessful authentication attempt with a wrong PACE passwords.
- FIA_UID.1/PACE_CAM, Timing of identification. The TOE allows to carry out the PACE CAM Protocol after successful user identification
- FIA_UAU.1/PACE_CAM, Timing of authentication. The TOE prevents reuse of authentication data related to the PACE CAM protocol, i.e. according authentication mechanisms.
- FIA_UAU.4/PACE_CAM, Single-use authentication mechanisms - Single-use authentication of the Terminal by the TOE
- FIA_UAU.5/PACE_CAM, Multiple authentication mechanisms to support user authentication. The TOE provides multiple authentication mechanisms, PACE_CAM, symmetric key based authentication mechanism, etc.
- FIA_UAU.6/PACE_CAM, Re-authenticating of Terminal by the TOE. The TOE re-authenticates the connected terminal, if a secure messaging error occurred.

The Secure messaging is ensured by:

- FCS_CKM.1/DH_PACE,FCS_CKM.1/DH_PACE_DH Diffie-Hellman key generation for PACE session keys is supported by SF.PACE. FCS_CKM.1/DH_PACE and FCS_CKM.1/DH_PACE_DH require that the TSF shall generate cryptographic keys based on ECDH compliant to ISO 15946 with specific domain parameters, meeting **[TR_03110]**, Annex A.1, or DH based on the Diffie-Hellman key derivation protocol compliant to PKCS#3 and **[TR_03110]**.
- FCS_CKM.4, Cryptographic key destruction – Session keys with support provided by SF.KEY_DESTRUCTION
- FCS_COP.1/PACE_ENC, Cryptographic operation – Encryption / Decryption AES / 3DES with support provided by SF.PACE

- FCS_COP.1/PACE_MAC, implements Cryptographic operation MAC/CMAC for PACE_CAM provided by SF.PACE

The TOE access control is ensured by

- FDP_ACF.1/FIREWALL, Security attribute based access control, to detect and block various failures or security violation is provided by SF_FIREWALL
- FDP_RIP.1/PACE, Subset residual information protection: this protection is provided by SF.PACE and used by PACE CAM module.
- FMT_MTD.1/PACE_CAM_KEY_WRITE, Modular invert of the CA key directly used for PACE-CAM.
- FMT_MTD.1/PACE_CAM_KEY_READ, Management of TSF data – Key Read protection of PACE Passwords and Modular invert of the CA key directly used for PACE-CAM: restricts the access to the Personalisation Agent Keys.

### *8.2.2 Association tables of SFRs and TSS*

| Security Functional Requirements | TOE Summary Specification |
|---|---|
| FCS_CKM.1 | SF_KEY_GENERATION |
| FCS_CKM.4 | SF_KEY_DESTRUCTION, SF.PACE_CAM - Authentication using PACE_CAM |
| FCS_COP.1 | SF_KEY_AGREEMENT, SF_MESSAGE_DIGEST, SF_ENCRYPTION_AND_DECRYPTION, SF_SIGNATURE |
| FDP_RIP.1/ABORT | SF_CLEARING_OF_SENSITIVE_INFORMATION, SF_ATOMIC_TRANSACTION |
| FDP_RIP.1/APDU | SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FDP_RIP.1/GlobalArray | SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FDP_RIP.1/bArray | SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FDP_RIP.1/KEYS | SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FDP_RIP.1/TRANSIENT | SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FDP_ROL.1/FIREWALL | SF_ATOMIC_TRANSACTION |
| FCS_RNG.1 | SF_SECURITY_FUNCTIONS_OF_THE_IC, SF_RANDOM_NUMBER |
| FDP_ACC.2/FIREWALL | SF_FIREWALL |
| FDP_ACF.1/FIREWALL | SF_FIREWALL, SF.PACE_CAM - Authentication using PACE_CAM |
| FDP_IFC.1/JCVM | SF_FIREWALL |
| FDP_IFF.1/JCVM | SF_FIREWALL |
| FDP_RIP.1/OBJECTS | SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FMT_MSA.1/JCRE | SF_FIREWALL |

| | |
|---|---|
| FMT_MSA.1/JCVM | SF_FIREWALL |
| FMT_MSA.2/FIREWALL_JCVM | SF_FIREWALL |
| FMT_MSA.3/FIREWALL | SF_FIREWALL |
| FMT_MSA.3/JCVM | SF_FIREWALL |
| FMT_SMF.1 | SF_CARD_CONTENT_MANAGEMENT |
| FMT_SMR.1 | SF_FIREWALL |
| FAU_ARP.1 | SF_FIREWALL, SF_EXCEPTION |
| FDP_SDI.2/DATA | SF_DATA_INTEGRITY |
| FPR_UNO.1 | SF_UNOBSERVABILITY |
| FPT_FLS.1 | SF_ATOMIC_TRANSACTION, SF_FIREWALL, SF_CARD_CONTENT_MANAGEMENT, SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FPT_TDC.1 | SF_CARD_MANAGEMENT_ENVIRONMENT |
| FIA_ATD.1/AID | SF_CARD_CONTENT_MANAGEMENT |
| FIA_UID.2/AID | SF_FIREWALL |
| FIA_USB.1/AID | SF_FIREWALL |
| FMT_MTD.1/JCRE | SF_CARD_CONTENT_MANAGEMENT |
| FMT_MTD.3/JCRE | SF_CARD_CONTENT_MANAGEMENT |
| FDP_ITC.2/Installer | SF_CARD_CONTENT_MANAGEMENT |
| FMT_SMR.1/Installer | SF_CARD_CONTENT_MANAGEMENT |
| FPT_FLS.1/Installer | SF_ATOMIC_TRANSACTION, SF_CARD_CONTENT_MANAGEMENT, SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FPT_RCV.3/Installer | SF_ATOMIC_TRANSACTION, SF_CARD_CONTENT_MANAGEMENT |
| FDP_ACC.2/ADEL | SF_CARD_CONTENT_MANAGEMENT |
| FDP_ACF.1/ADEL | SF_CARD_CONTENT_MANAGEMENT |
| FDP_RIP.1/ADEL | SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FMT_MSA.1/ADEL | SF_CARD_CONTENT_MANAGEMENT |
| FMT_MSA.3/ADEL | SF_CARD_CONTENT_MANAGEMENT |
| FMT_SMF.1/ADEL | SF_CARD_MANAGEMENT_ENVIRONMENT |
| FMT_SMR.1/ADEL | SF_CARD_MANAGEMENT_ENVIRONMENT |
| FPT_FLS.1/ADEL | SF_CARD_CONTENT_MANAGEMENT, SF_ATOMIC_TRANSACTION, SF_CLEARING_OF_SENSITIVE_INFORMATION |

| | |
|---|---|
| FDP_RIP.1/ODEL | SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FPT_FLS.1/ODEL | SF_CLEARING_OF_SENSITIVE_INFORMATION |
| FCO_NRO.2/CM | SF_CARD_CONTENT_MANAGEMENT |
| FDP_IFC.2/CM | SF_CARD_CONTENT_MANAGEMENT, SF_DAP_VERIFICATION |
| FDP_IFF.1/CM | SF_CARD_CONTENT_MANAGEMENT, SF_DAP_VERIFICATION |
| FDP_UIT.1/CM | SF_DAP_VERIFICATION, SF_ENCRYPTION_AND_DECRYPTION |
| FIA_UID.1/CM | SF_GP_DISPATCHER |
| FMT_MSA.1/CM | SF_CARD_CONTENT_MANAGEMENT, SF_KEY_MANAGEMENT |
| FMT_MSA.3/CM | SF_CARD_CONTENT_MANAGEMENT |
| FMT_SMF.1/CM | SF_CARD_CONTENT_MANAGEMENT |
| FMT_SMR.1/CM | SF_CARD_CONTENT_MANAGEMENT |
| FTP_ITC.1/CM | SF_CARD_CONTENT_MANAGEMENT, SF_DAP_VERIFICATION, SF_GP_DISPATCHER |
| FPT_TST.1 | SF_HARDWARE_OPERATING, SF_DATA_INTEGRITY, SF.PACE - Authentication using PACE |
| FCO_NRO.2/CM_DAP | SF_DAP_VERIFICATION |
| FIA_AFL.1/CM | SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL |
| FIA_UAU.1/CM | SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL, SF_GP_DISPATCHER |
| FIA_UAU.4/CardIssuer | SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL, SF_RANDOM_NUMBER |
| FIA_UAU.7/CardIssuer | SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL, SF_RANDOM_NUMBER |
| FPR_UNO.1/Key_CM | SF_KEY_MANAGEMENT, SF_UNOBSERVABILITY |
| FPT_TDC.1/CM | SF_KEY_MANAGEMENT |
| FMT_SMR.2/CM | SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL |
| FCS_COP.1/CM | SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL, SF_ENCRYPTION_AND_DECRYPTION |
| FDP_ACC.2/PP | SF_MANUFACTURER_AUTHENTICATION, SF_RESIDENT_APPLICATION_DISPATCHER |
| FDP_ACF.1/PP | SF_MANUFACTURER_AUTHENTICATION |

| | |
|---|---|
| FDP_UCT.1/PP | SF_MANUFACTURER_AUTHENTICATION, SF_PREPERSO_AND_PATCHING, SF_RESIDENT_APPLICATION_DISPATCHER |
| FDP_ITC.1/PP | SF_MANUFACTURER_AUTHENTICATION, SF_RESIDENT_APPLICATION_DISPATCHER, SF_PREPERSO_AND_PATCHING |
| FIA_AFL.1/PP | SF_MANUFACTURER_AUTHENTICATION |
| FIA_UAU.1/PP | SF_RESIDENT_APPLICATION_DISPATCHER |
| FIA_UID.1/PP | SF_RESIDENT_APPLICATION_DISPATCHER |
| FMT_MSA.1/PP | SF_MANUFACTURER_AUTHENTICATION |
| FMT_SMF.1/PP | SF_MANUFACTURER_AUTHENTICATION |
| FIA_UAU.4/CardManu | SF_MANUFACTURER_AUTHENTICATION |
| FIA_UAU.7/CardManu | SF_MANUFACTURER_AUTHENTICATION |
| FMT_MOF.1/PP | SF_PREPERSO_AND_PATCHING, SF_CARD_MANAGEMENT_ENVIRONMENT |
| FMT_SMR.2/PP | SF_MANUFACTURER_AUTHENTICATION, SF_PREPERSO_AND_PATCHING, SF_RESIDENT_APPLICATION_DISPATCHER |
| FMT_MSA.3/PP | SF_MANUFACTURER_AUTHENTICATION, SF_PREPERSO_AND_PATCHING |
| FCS_COP.1/PP | SF_ENCRYPTION_AND_DECRYPTION, SF_MANUFACTURER_AUTHENTICATION, SF_PREPERSO_AND_PATCHING |
| FCS_CKM.4/PP | SF_KEY_DESTRUCTION |
| FDP_UIT.1/PP | SF_PREPERSO_AND_PATCHING |
| FAU_STG.2 | SF_PREPERSO_AND_PATCHING |
| FPT_PHP.3/SCP | SF_HARDWARE_OPERATING |
| FPT_RCV.4/SCP | SF_DATA_COHERENCY |
| FRU_FLT.1/SCP | SF_MEMORY_FAILURE |
| FPR_UNO.1/USE_KEY | SF_UNOBSERVABILITY |
| FIA_AFL.1/PIN | SF_CARDHOLDER_VERIFICATION |
| FMT_MTD.2/GP_PIN | SF_CARDHOLDER_VERIFICATION |
| FMT_MTD.1/PIN | SF_CARDHOLDER_VERIFICATION |
| FIA_AFL.1/GP_PIN | SF_CARDHOLDER_VERIFICATION |
| FDP_ACC.2/RV_Stack | SF_RUNTIME_VERIFIER |
| FDP_ACF.1/RV_Stack | SF_RUNTIME_VERIFIER |

| | |
|---|---|
| FMT_MSA.1/RV_Stack | SF_RUNTIME_VERIFIER |
| FMT_MSA.2/RV_Stack | SF_RUNTIME_VERIFIER |
| FMT_MSA.3/RV_Stack | SF_RUNTIME_VERIFIER |
| FMT_SMF.1/RV_Stack | SF_RUNTIME_VERIFIER |
| FDP_ACC.2/RV_Heap | SF_RUNTIME_VERIFIER |
| FDP_ACF.1/RV_Heap | SF_RUNTIME_VERIFIER |
| FMT_MSA.1/RV_Heap | SF_RUNTIME_VERIFIER |
| FMT_MSA.2/RV_Heap | SF_RUNTIME_VERIFIER |
| FMT_MSA.3/RV_Heap | SF_RUNTIME_VERIFIER |
| FMT_SMF.1/RV_Heap | SF_RUNTIME_VERIFIER |
| FDP_ACC.2/RV_Transient | SF_RUNTIME_VERIFIER |
| FDP_ACF.1/RV_Transient | SF_RUNTIME_VERIFIER |
| FMT_MSA.1/RV_Transient | SF_RUNTIME_VERIFIER |
| FMT_MSA.2/RV_Transient | SF_RUNTIME_VERIFIER |
| FMT_MSA.3/RV_Transient | SF_RUNTIME_VERIFIER |
| FMT_SMF.1/RV_Transient | SF_RUNTIME_VERIFIER |
| FDP_ACC.2/JBox | SF_JBOX |
| FDP_ACF.1/JBox | SF_JBOX |
| FMT_MSA.1/JBox | SF_JBOX |
| FMT_MSA.3/JBox | SF_JBOX |
| FMT_SMF.1/JBox | SF_JBOX |
| FMT_MTD.1/Activate_DBI | SF_ATOMIC_TRANSACTION, SF_FIREWALL, SF_CARD_MANAGEMENT_ENVIRONMENT |
| FMT_MTD.1/Deactivate_DBI | SF_ATOMIC_TRANSACTION, SF_FLEXICODE, SF_CARD_MANAGEMENT_ENVIRONMENT, SF_FIREWALL |
| FMT_MOF.1/FlxC | SF_FLEXICODE, SF_MANUFACTURER_AUTHENTICATION, SF_RESIDENT_APPLICATION_DISPATCHER |
| FMT_SMF.1/FlxC | SF_FLEXICODE |
| FDP_ACC.2/FlxC | SF_MANUFACTURER_AUTHENTICATION, SF_RESIDENT_APPLICATION_DISPATCHER, SF_FLEXICODE |
| FDP_ACF.1/FlxC | SF_MANUFACTURER_AUTHENTICATION, SF_FLEXICODE |

| | |
|---|---|
| FMT_MSA.1/FlxC | SF_FLEXICODE, SF_MANUFACTURER_AUTHENTICATION, SF_PREPERSO_AND_PATCHING |
| FMT_MSA.2/FlxC | SF_FLEXICODE, SF_PREPERSO_AND_PATCHING, SF_MANUFACTURER_AUTHENTICATION |
| FMT_MSA.3/FlxC | SF_FLEXICODE, SF_MANUFACTURER_AUTHENTICATION, SF_PREPERSO_AND_PATCHING |
| FMT_MSA.4/FlxC | SF_FLEXICODE, SF_PREPERSO_AND_PATCHING, SF_MANUFACTURER_AUTHENTICATION |
| FPT_FLS.1/FlxC | SF_FLEXICODE |
| FDP_RIP.1/FlxC | SF_FLEXICODE |
| FDP_SDI.2/MONOTONIC_COUNTER | SF_DATA_INTEGRITY |
| FDP_SDI.2/ARRAY | SF_DATA_INTEGRITY |
| FCS_CKM.1/DH_PACE | SF.PACE - Authentication using PACE, SF.PACE_CAM - Authentication using PACE_CAM |
| FCS_COP.1/PACE_ENC | SF.PACE - Authentication using PACE, SF.PACE_CAM - Authentication using PACE_CAM |
| FCS_COP.1/PACE_MAC | SF.PACE - Authentication using PACE, SF.PACE_CAM - Authentication using PACE_CAM |
| FCS_CKM.4/DH_PACE | SF.PACE - Authentication using PACE |
| FCS_RND.1/PACE | SF.PACE - Authentication using PACE |
| FIA_AFL.1/PACE | SF.PACE - Authentication using PACE, SF.PACE_CAM - Authentication using PACE_CAM |
| FIA_UID.1/PACE | SF.PACE - Authentication using PACE |
| FIA_UAU.1/PACE | SF.PACE - Authentication using PACE |
| FIA_UAU.4/PACE | SF.PACE - Authentication using PACE |
| FIA_UAU.5/PACE | SF.PACE - Authentication using PACE |
| FIA_UAU.6/PACE | SF.PACE - Authentication using PACE |
| FTP_ITC.1/PACE | SF.PACE - Authentication using PACE |
| FMT_SMR.1/PACE | SF.PACE - Authentication using PACE |
| FMT_SMF.1/PACE | SF.PACE - Authentication using PACE |
| FMT_LIM.1/PACE_Perso | SF.PACE - Authentication using PACE |
| FMT_LIM.2/PACE_Perso | SF.PACE - Authentication using PACE |
| FMT_MTD.1/INI_ENA | SF.PACE - Authentication using PACE |
| FMT_MTD.1/INI_DIS | SF.PACE - Authentication using PACE |

| | |
|---|---|
| FMT_MTD.1/KEY_READ | SF.PACE - Authentication using PACE |
| FDP_RIP.1/PACE | SF.PACE - Authentication using PACE, SF.PACE_CAM - Authentication using PACE_CAM |
| FPT_EMS.1/PACE | SF.PACE - Authentication using PACE |
| FIA_UID.1/PACE_CAM | SF.PACE_CAM - Authentication using PACE_CAM |
| FIA_UAU.1/PACE_CAM | SF.PACE_CAM - Authentication using PACE_CAM |
| FIA_UAU.4/PACE_CAM | SF.PACE_CAM - Authentication using PACE_CAM |
| FIA_UAU.5/PACE_CAM | SF.PACE_CAM - Authentication using PACE_CAM |
| FIA_UAU.6/PACE_CAM | SF.PACE_CAM - Authentication using PACE_CAM |
| FMT_MTD.1/PACE_CAM_KEY_READ | SF.PACE_CAM - Authentication using PACE_CAM |
| FMT_MTD.1/PACE_CAM_KEY_WRITE | SF.PACE_CAM - Authentication using PACE_CAM |
| FCS_CKM.1/RSA | SF_KEY_GENERATION |
| FCS_COP.1/RSA | SF_KEY_AGREEMENT, SF_MESSAGE_DIGEST, SF_ENCRYPTION_AND_DECRYPTION, SF_SIGNATURE |
| FCS_COP.1/SHA3 | SF_KEY_AGREEMENT, SF_MESSAGE_DIGEST, SF_ENCRYPTION_AND_DECRYPTION, SF_SIGNATURE |
| FIA_AFL.1/BIO | SF_DATA_INTEGRITY, SF_CARDHOLDER_VERIFICATION |
| FMT_MTD.1/BIO | SF_ATOMIC_TRANSACTION, SF_DATA_INTEGRITY, SF_KEY_MANAGEMENT, SF_CARDHOLDER_VERIFICATION |
| FCS_CKM.1/DH_PACE_DH | SF.PACE - Authentication using PACE, SF.PACE_CAM - Authentication using PACE_CAM, SF_FLEXICODE |
| FCS_COP.1/HMAC | SF_SIGNATURE, SF_MESSAGE_DIGEST, SF_ENCRYPTION_AND_DECRYPTION |

**Table 15 SFRs and TSS - Coverage**

| TOE Summary Specification | Security Functional Requirements |
|---|---|
| SF_ATOMIC_TRANSACTION | FPT_FLS.1/Installer, FPT_RCV.3/Installer, FPT_FLS.1/ADEL, FMT_MTD.1/BIO, FDP_RIP.1/ABORT, FDP_ROL.1/FIREWALL, FPT_FLS.1, FMT_MTD.1/Activate_DBI, FMT_MTD.1/Deactivate_DBI |
| SF_UNOBSERVABILITY | FPR_UNO.1, FPR_UNO.1/Key_CM, FPR_UNO.1/USE_KEY |

| | |
|---|---|
| SF_SIGNATURE | FCS_COP.1/RSA, FCS_COP.1/SHA3, FCS_COP.1/HMAC, FCS_COP.1 |
| SF_SECURITY_FUNCTIONS_OF_THE_IC | FCS_RNG.1 |
| SF_RUNTIME_VERIFIER | FDP_ACC.2/RV_Stack, FDP_ACF.1/RV_Stack, FMT_MSA.1/RV_Stack, FMT_MSA.2/RV_Stack, FMT_MSA.3/RV_Stack, FMT_SMF.1/RV_Stack, FDP_ACC.2/RV_Heap, FDP_ACF.1/RV_Heap, FMT_MSA.1/RV_Heap, FMT_MSA.2/RV_Heap, FMT_MSA.3/RV_Heap, FMT_SMF.1/RV_Heap, FDP_ACC.2/RV_Transient, FDP_ACF.1/RV_Transient, FMT_MSA.1/RV_Transient, FMT_MSA.2/RV_Transient, FMT_MSA.3/RV_Transient, FMT_SMF.1/RV_Transient |
| SF_RESIDENT_APPLICATION_DISPATCHER | FMT_MOF.1/FlxC, FDP_ACC.2/FlxC, FCS_COP.1/CM, FDP_ACC.2/PP, FDP_UCT.1/PP, FDP_ITC.1/PP, FIA_UAU.1/PP, FIA_UID.1/PP, FMT_SMR.2/PP |
| SF_FLEXICODE | FMT_MOF.1/FlxC, FMT_SMF.1/FlxC, FDP_ACC.2/FlxC, FDP_ACF.1/FlxC, FMT_MSA.1/FlxC, FMT_MSA.2/FlxC, FMT_MSA.3/FlxC, FMT_MSA.4/FlxC, FPT_FLS.1/FlxC, FDP_RIP.1/FlxC, FCS_CKM.1/DH_PACE_DH, FMT_MTD.1/Deactivate_DBI |
| SF_RANDOM_NUMBER | FCS_RNG.1, FIA_UAU.4/CardIssuer, FIA_UAU.7/CardIssuer |
| SF_PREPERSO_AND_PATCHING | FMT_MSA.1/FlxC, FMT_MSA.2/FlxC, FMT_MSA.3/FlxC, FMT_MSA.4/FlxC, FDP_UCT.1/PP, FDP_ITC.1/PP, FMT_MOF.1/PP, FMT_SMR.2/PP, FMT_MSA.3/PP, FCS_COP.1/PP, FDP_UIT.1/PP, FAU_STG.2 |
| SF_MEMORY_FAILURE | FRU_FLT.1/SCP |
| SF_MESSAGE_DIGEST | FCS_COP.1/RSA, FCS_COP.1/SHA3, FCS_COP.1/HMAC, FCS_COP.1 |

| | |
|---|---|
| SF_MANUFACTURER_AUTHENTICATION | FMT_MOF.1/FlxC, FDP_ACC.2/FlxC, FDP_ACF.1/FlxC, FMT_MSA.1/FlxC, FMT_MSA.2/FlxC, FMT_MSA.3/FlxC, FMT_MSA.4/FlxC, FDP_ACC.2/PP, FDP_ACF.1/PP, FDP_UCT.1/PP, FDP_ITC.1/PP, FIA_AFL.1/PP, FMT_MSA.1/PP, FMT_SMF.1/PP, FIA_UAU.4/CardManu, FIA_UAU.7/CardManu, FMT_SMR.2/PP, FMT_MSA.3/PP, FCS_COP.1/PP |
| SF_KEY_MANAGEMENT | FMT_MTD.1/BIO, FMT_MSA.1/CM, FPR_UNO.1/Key_CM, FPT_TDC.1/CM |
| SF_KEY_GENERATION | FCS_CKM.1/RSA, FCS_CKM.1 |
| SF_KEY_DESTRUCTION | FCS_CKM.4, FCS_CKM.4/PP |
| SF_KEY_AGREEMENT | FCS_COP.1/RSA, FCS_COP.1/SHA3, FCS_COP.1 |
| SF_JBOX | FDP_ACC.2/JBox, FDP_ACF.1/JBox, FMT_MSA.1/JBox, FMT_MSA.3/JBox, FMT_SMF.1/JBox |
| SF_HARDWARE_OPERATING | FPT_TST.1, FDP_UIT.1/PP, FPT_PHP.3/SCP |
| SF_GP_DISPATCHER | FIA_UID.1/CM, FTP_ITC.1/CM, FIA_UAU.1/CM |
| SF_FIREWALL | FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_MSA.2/FIREWALL_JCVM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_SMR.1, FAU_ARP.1, FPT_FLS.1, FIA_UID.2/AID, FIA_USB.1/AID, FMT_MTD.1/Activate_DBI, FMT_MTD.1/Deactivate_DBI |
| SF_EXCEPTION | FAU_ARP.1 |
| SF_ENTITY_AUTHENTICATION/SECURE_CHANNEL | FIA_AFL.1/CM, FIA_UAU.1/CM, FIA_UAU.4/CardIssuer, FIA_UAU.7/CardIssuer, FMT_SMR.2/CM, FCS_COP.1/CM |
| SF_ENCRYPTION_AND_DECRYPTION | FCS_COP.1/RSA, FCS_COP.1/SHA3, FCS_COP.1, FDP_UIT.1/CM, FCS_COP.1/CM, FCS_COP.1/PP |

| | |
|---|---|
| SF_DATA_INTEGRITY | FDP_SDI.2/MONOTONIC_COUNTER, FDP_SDI.2/ARRAY, FIA_AFL.1/BIO, FMT_MTD.1/BIO, FDP_SDI.2/DATA, FPT_TST.1 |
| SF_DATA_COHERENCY | FPT_RCV.4/SCP |
| SF_DAP_VERIFICATION | FDP_IFC.2/CM, FDP_IFF.1/CM, FDP_UIT.1/CM, FTP_ITC.1/CM, FCO_NRO.2/CM_DAP |
| SF_CLEARING_OF_SENSITIVE_INFORMATION | FPT_FLS.1/Installer, FDP_RIP.1/ADEL, FPT_FLS.1/ADEL, FDP_RIP.1/ODEL, FPT_FLS.1/ODEL, FDP_RIP.1/ABORT, FDP_RIP.1/APDU, FDP_RIP.1/GlobalArray, FDP_RIP.1/bArray, FDP_RIP.1/KEYS, FDP_RIP.1/TRANSIENT, FDP_RIP.1/OBJECTS, FPT_FLS.1 |
| SF_CARDHOLDER_VERIFICATION | FIA_AFL.1/BIO, FMT_MTD.1/BIO, FIA_AFL.1/PIN, FMT_MTD.2/GP_PIN, FMT_MTD.1/PIN, FIA_AFL.1/GP_PIN |
| SF_CARD_MANAGEMENT_ENVIRONMENT | FMT_SMF.1/ADEL, FMT_SMR.1/ADEL, FPT_TDC.1, FMT_MOF.1/PP, FMT_MTD.1/Activate_DBI, FMT_MTD.1/Deactivate_DBI |
| SF_CARD_CONTENT_MANAGEMENT | FDP_ITC.2/Installer, FMT_SMR.1/Installer, FPT_FLS.1/Installer, FPT_RCV.3/Installer, FDP_ACC.2/ADEL, FDP_ACF.1/ADEL, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FPT_FLS.1/ADEL, FMT_SMF.1, FPT_FLS.1, FIA_ATD.1/AID, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FCO_NRO.2/CM, FDP_IFC.2/CM, FDP_IFF.1/CM, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMF.1/CM, FMT_SMR.1/CM, FTP_ITC.1/CM |
| SF.PACE - Authentication using PACE | FCS_CKM.1/DH_PACE_DH, FPT_TST.1, FCS_CKM.1/DH_PACE, FCS_COP.1/PACE_ENC, FCS_COP.1/PACE_MAC, FCS_CKM.4/DH_PACE, FCS_RND.1/PACE, FIA_AFL.1/PACE, FIA_UID.1/PACE, FIA_UAU.1/PACE, FIA_UAU.4/PACE, FIA_UAU.5/PACE, FIA_UAU.6/PACE, FTP_ITC.1/PACE, FMT_SMR.1/PACE, FMT_SMF.1/PACE, |

| | |
|---|---|
| | FMT_LIM.1/PACE_Perso, FMT_LIM.2/PACE_Perso, FMT_MTD.1/INI_ENA, FMT_MTD.1/INI_DIS, FMT_MTD.1/KEY_READ, FDP_RIP.1/PACE, FPT_EMS.1/PACE |
| SF.PACE_CAM - Authentication using PACE_CAM | FIA_UID.1/PACE_CAM, FIA_UAU.1/PACE_CAM, FIA_UAU.4/PACE_CAM, FIA_UAU.5/PACE_CAM, FIA_UAU.6/PACE_CAM, FMT_MTD.1/PACE_CAM_KEY_READ, FMT_MTD.1/PACE_CAM_KEY_WRITE, FCS_CKM.1/DH_PACE_DH, FCS_CKM.4, FDP_ACF.1/FIREWALL, FCS_CKM.1/DH_PACE, FCS_COP.1/PACE_ENC, FCS_COP.1/PACE_MAC, FIA_AFL.1/PACE, FDP_RIP.1/PACE |

**Table 16 TSS and SFRs - Coverage**

# 9 INDEX

**O**

**U**