MultiApp V5.2: JCS

Common Criteria / ISO 15408
Security Target – Public version
EAL6+

Version 1.7p – September 12th 2025

CONTENT

FIGURES

TABLES

1 SECURITY TARGET INTRODUCTION

1.1 SECURITY TARGET REFERENCE

Title :	MultiApp V5.2: JCS Security Target
Version :	1.7p
ST Reference :	D1593231
Origin :	Thales
IT Security Evaluation Facility :	LETI
IT Security Certification scheme :	Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)

1.2 TOE REFERENCE

Product Technical Name :	MultiApp V5.2
Product Commercial Names:	MultiApp 5.2 Premium PQC
Security Controllers :	SLC38GDA800 (INFINEON IFX_CCI_000043h)
TOE Name :	MultiApp 5.2 Premium PQC
TOE Version :	5.2 (OS Release Date 0x5114)
TOE documentation :	Guidance [AGD]
Composition elements:	
Composite TOE identifier:	IFX_CCI_000043h
Composite TOE Version:	**Hardware** Version: S11 **Firmware** *BOS & POWS: Version 80.309.05.0* *Flash Loader: Version 09.13.0004* **Software** *NRG™ SW (optional) 05.03.4097* *HSL (optional) v3.52.9708* *UMSLC v01.30.0564* *SCL (optional) v2.15.000* *ACL (optional) v3.33.003 and v3.34.000 and v3.35.001* *RCL (optional) v1.10.007* *HCL (optional) v1.13.002*

1.3 TOE IDENTIFICATION

The TOE identification is provided by the Tag identity and CPLC data. These data are available by executing a dedicated command described in [AGD-OPE] and here below:

The TOE can be identified through the Get Data Command response with tag "0103", as follows:

Name	Length	Description	Value	Participate to TOE identification
Thales Family Name	1	Java Card	0xB0	YES
Thales OS name	1	MultiApp	0x85	YES
Thales Mask Number	1	MultiappV5.2	0x69	YES
Thales Product Name	1		0x6C	YES
Flow id Version	1		0x01	YES
Filter set	1		0x00	YES
Chip Manufacturer	2	Infineon	0x4090	YES
Chip Identifier	2	Master chip Identifier	0x9202 (SLU38IME800A4-S11) Non-exhaustive values	NO
BPU	2	BPU configuration	SLC38G CA600 – 0x8004 – Contact only 600k SLC38G CA800 – 0x8006 – Contact only 800k SLC38G DA600 – 0x8007 – Dual 600k with VHBR SLC38G DA600A8 – 0x8010 – Dual 600k wo VHBR SLC38G DA800 – **0x8019** – Dual 800k with VHBR SLC38G DA800A8 – 0x8022 – Dual 800k wo VHBR Non-exhaustive values	NO
PDM TP	3			NO
PDM CI	3			NO
Feature Flag – Crypto Config	2	See after		NO
Feature Flag – Feature Config byte 1	1	See after		NO
Feature Flag – Feature Config byte 2	1	See after		NO
Platform Certificates	1		Bit 7 (0x40): CC Configuration	YES (only for bit 7)
APPLI CERTIFICATES byte 1	1		Bit 8 (0x80): eTravel Bit 7 (0x40): IAS Classic Bit 6 (0x20) : Reserved Bit 5 (0x10) : Reserved Bit 4 (0x08) : Q-IAS Bit 3-1 : Not used (0)	NO (only for bit 8 & 7 & 4) to be adapted according to the applet(s) installed
APPLI CERTIFICATES byte 2	1		00h	NO

Note: the eight first fields of this table (from “Thales Family Name” to “Chip Identifier”) are used for traceability purpose.

Also, using Get data command with tag 9F7F for product identification:

Name	length	Description	Value	Participate to TOE identification
IC Fabricator	2	Chip fabricator	0x40 0x90	YES
IC Type	2	Chip model number	0x00 0x43	YES
Operating system identifier	2	OS developer	0x19 0x81	YES
Operating system release date	2	Date reference	0x51 0x14	YES
Operating system release level	2	5.2	0x05 0x20	YES

The TOE and the product differ, as further explained in **Product Architecture**

- The TOE is the JCS open platform of the MultiApp V5.2 product.
- The MultiApp V5.2 product also includes applets.

Optional features / Field (extract from identity tag)	Crypto features byte A								Crypto features byte B								features byte 2								features byte 1							
bit	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
ECC								X																								
HMAC					X																											
RSA																X																
RSA-DH													X																			
RSA-OBKG												X																				
ML-DSA-65										X																						
ML-KEM (KYBER)									X																							
PACE DH																							X									
PACE ECC																						X										
File system																					X											
ISM																				X												
Etravel																			X													
EAC/GAP																		X														
Linker																	X															
Biometry Fingerprint																																X
Biometry Facial																															X	
Biometry IRIS																														X		
FIPS																																

Table 1: MAV 5.2 Features configuration

Note 1: X with value 1 when the feature is available, X with value 0 when the feature is not available.
Note 2: The bits that are not listed in the table 1 are considered as RFU

1.4 SECURITY TARGET OVERVIEW

The main objectives of this ST are:

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

1.5 REFERENCES

1.5.1 External References

[CC]	Common Criteria references
[CC-1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017.
[CC-2]	Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, CCMB-2017-04-002, Version 3.1 Revision 5, April 2017.
[CC-3]	Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.
[CEM]	Common Methodology for Information Technology Security Evaluation Methodology CCMB-2017604-001, version 3.1 rev 5 April 2017
[JIL_CPE]	Joint Interpretation Library: Composite product evaluation for Smart Cards and similar devices, Version 1.5.1 May 2018
[PP]	**Protection profiles**
[PP-IC-0084]	Security IC Platform Protection Profile with augmentation Packages– BSI-CC-PP-0084-2014 version 1.0
[PP-JCS-Open]	Java Card System Protection Profile – Open Configuration BSI-CC-PP-0099-V2-2020, Version 3.1, April 2020
[RGS-B1]	ANSSI, « Référentiel général de sécurité », https://cyber.gouv.fr/sites/default/files/2021/03/anssi-guide-mecanismes_crypto-2.04.pdf Annexe B1 Mécanismes cryptographiques, règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques; version 2.0.4, 2020-01-01
[AIS31]	A proposal for: Functionality classes for random number generators Version 2.0 Sept 2011
[PP_PACE]	Protection Profile, Machine Readable Travel Document using Standard Inspection Procedure with PACE, version 1.01, 22 July 2014. Certified and maintained by the BSI (Bundesamt für Sicherheit in der Informationstechnik) under the reference BSI-CC-PP-0068-V2-2011-MA-01.
[PP_EAC2]	Protection Profile, Machine Readable Travel Document with “ICAO Application”, Extended Access Control with PACE, version 1.3.2, 2012, December 5th. Certified and maintained by BSI (Bundesamt für Sicherheit in der Informationstechnik) under reference BSI-PP-0056-V2-MA-2012
[PP_BAC]	"Protection Profile, Machine Readable Travel Document with “ICAO Application”, Basic Access Control, version 1.10, 25 March 2009. Certified and maintained by the BSI (Bundesamt für Sicherheit in der Informationstechnik) under the reference BSI-PP-0055-2009.
[INFINEON]	**Thales Design Services References**
[IFX-IC]	IFX_CCI_00003Bh, IFX_CCI_000043h, IFX_CCI_00005Dh, IFX_CCI_00005Eh, IFX_CCI_00005Fh, IFX_CCI_000060h, IFX_CCI_000061h, IFX_CCI_000062h, IFX_CCI_000063h, IFX_CCI_000064h S11 Security Target Lite Revision: V5.9
[CR-IC]	BSI-DSZ-CC-1169-V4-2024 Date 13 Septembre 2024
[NIST]	**NIST references**
[FIPS180-2]	*Federal Information Processing Standards Publication 180-2 SECURE HASH STANDARD (+Change Notice to include SHA-224),* U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, 2002 August 1
[FIPS197]	*Federal Information Processing Standards Publication 197 ADVANCED ENCRYPTION STANDARD (AES),* 2001 November 26

[SP800-67]	NIST Special Publication 800-67 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, Revision 1 – Revised January 2012.
[FIPS202]	Federal Information Processing Standards Publication SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August 2015
[FIPS180-4]	Federal Information Processing Standards Publication Secure Hash Standard (SHS), August 2015
[ISO]	**ISO references**
[ISO9796-2]	*ISO/IEC 9796-2:2010: Information technology – Security techniques – Digital Signature Schemes giving message recovery – Part 2: Integer factorization based mechanisms,* Third edition 2010-12-15
[ISO9797-1]	*ISO/IEC 9797-1:2011: Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher,* Second edition 2011-03-01
[GP]	**Global Platform references**
[GP23]	GP2.3.1: GPC_CardSpecification_v2.3.1_PublicRelease_CC.pdf GlobalPlatform Technology Card Specification Version 2.3.1 - March 2018 Reference: GPC_SPE_034
[GP23 Amend D]	Card Technology Secure Channel Protocol '03' Card Specification v2.3 – Amendment D Version 1.1.2 - March 2019 Reference: GPC_SPE_014
[GP23 Amend E]	Card Technology Security Upgrade for Card Content Management Card Specification v2.3 – Amendment E Version 1.1 - October 2016 Reference: GPC_SPE_042
[GP23 Com]	Global Platform – Card Common Implementation Configuration Version v2.1 - July 2018
[GP PF]	GlobalPlatform TechnologyCard Specification – Privacy FrameworkVersion 1.0.1 Public Release November 2019 Document Reference: GPC_SPE_10
[Others]	**Others specification references**
[TR03110-1]	Technical Guideline TR-03110-1 Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token – Part 1 – eMRTDs with BAC/PACEv2 and EACv1 Version 2.20, 26/02/2015
[TR03110-2]	Technical Guideline TR-03110 Advanced Security Mechanisms for Machine Readable Travel Documents and eIDAS Token – Part 2: Protocols for electronic IDentification, Authentication and trust Services (eIDAS) Version 2.21, 21/12/2016
[TRSIGN]	Technical report : Signature creation and administration for eIDAS token:Part 1: Functional Specification, Version 1.0, 2015/07/21 (ANSSI/BSI technical specifications)
[ICAO-TR-SAC]	ICAO TR – Supplemental Access Control for Machine Readable Travel Document, Version 1.1, April 15, 2014.
[ICAO-9303]	International Civil Aviation Organization, ICAO Doc 9303, Machine Readable Travel Documents – Machine Readable Passports, Version Sixth Edition, 2006 (this includes the latest supplemental for ICAO Doc 9303 which also should be considered)

[JCS]	**Javacard references**
[JAVASPEC]	The Java Language Specification. Third Edition, May 2005. Gosling, Joy, Steele and Bracha. ISBN 0-321-24678-0.
[JVM]	The Java Virtual Machine Specification. Lindholm, Yellin. ISBN 0-201-43294-3.
[JCBV]	Java Card Platform, version 2.2 Off-Card Verifier. June 2002. White paper. Published by Sun Microsystems, Inc.

[JCRE3]	Java Card 3.2 Runtime Environment (JCRE) Specification – January 2023 – Published by Oracle
[JCVM3]	Java Card 3.2 Virtual Machine (JCVM) Specification – January 2023 – Published by Oracle
[JCAPI3]	Java Card 3.2 Application Programming Interface (API) Specification, Classic Edition - January 2023 – Published by Oracle

1.5.2 Internal References [IR]

[AGD]	MulitiApp V5.2 Software – Guidance documentation
[AGD-PRE]	Preparation Guidance, D1600885
[AGD-OPE]	Operational Guidance, D1600884
[AGD-Ref]	MultiApp ID V5 Operating System – Reference Manual, D1525385D
[Applet guidance]	MultiApp Guidance Document - Guidance document for secure development for MultiApp products, D1539156
[PQC_GUIDE]	Guide for CC certified PQC Signatures, D1610996
[SPM]	MulitiApp V5.2Software – Security Policy Modeling
[MAV52_SPM]	MultiApp V5.2: ADV_SPM.1, D1607505

1.6 ACRONYMS AND GLOSSARY

AES	Advanced Encryption Standard
APDU	Application Protocol Data Unit
API	Application Programming Interface
CAD	Card Acceptance Device
CC	Common Criteria
CPU	Central Processing Unit
DES	Data Encryption Standard
DRNG	Deterministic Random Number Generator
EAL	Evaluation Assurance Level
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EEPROM	Electrically-Erasable Programmable Read-Only Memory
ES	Embedded Software
GP	Global Platform
IC	Integrated Circuit
IT	Information Technology
JCRE	JavaCard Runtime Environment
JCS	JavaCard System
JCVM	JavaCard Virtual Machine
ML-DSA	Module-Lattice-Based Digital Signature Well known as Crystals-Dilithium,this is the standard for PQC algorithm. Reference to NIST standard FIPS 204. Crystal-Dilithium level 2 = ML-DSA-44 Crystal-Dilithium level 3 = ML-DSA-65 Crystal-Dilithium level 5 = ML-DSA-87
ML-KEM	Module-Lattice-Based Key-Encapsulation Mechanism Standard. It is derived from CRYSTALS-KYBER. Reference to NIST standard FIPS 203.
NVM	Non-Volatile Memory
OP	Open Platform
PIN	Personal Identification Number
PP	Protection Profile
PQC	Post Quantum Cryptography
RMI	Remote Method Invocation
RNG	Random Number Generator
ROM	Read-Only Memory
RSA	Rivest Shamir Adleman
SAR	Security Assurance Requirement
SC	Smart Card
SCP	Secure Channel Protocol

SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
ST	Security Target
TOE	Target Of Evaluation
TSF	TOE Security Functionality

2 TOE OVERVIEW

2.1 TOE TYPE

The Java Card technology combines a subset of the Java programming language with a runtime environment optimized for smart cards and similar small-memory embedded devices [JCVM3]. The Java Card platform is a smart card platform enabled with Java Card technology (also called a “Java card”). This technology allows for multiple applications to run on a single card and provides facilities for secure interoperability of applications. Applications for the Java Card platform (“Java Card applications”) are called applets.

This TOE provides the security of an EAL6+ evaluated card with the flexibility of an open platform.
It allows for the loading of applets before or after the issuance of the card. These applets MAY or MAY NOT be evaluated on this platform.

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

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

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

2.2 PRODUCT ARCHITECTURE

The TOE is part of the *MultiApp V5.2* smartcard product. This smartcard contains the software dedicated to the operation of:

- The MultiApp V5.2 Platform, which supports the execution of the personalized applets and provides the smartcard administration services. It is conformant to Java Card 3.2 and GP 2.3.1 standards [GP23]. (With common configuration 2.1 [GP23 Com]) and with GP Privacy Framework v1.0.1 [GP PF].The identity applets: GDP v3.0, IAS classic V5.2.3, eTravel v3.2, BioPin Managemer v3.1 (MOCA server/client), MPCOS v4.1, MSFT PnP v1.0, FIDO Authentificator v2.1 applet, LDSv2 v1.1 , PURE DI v3.05, Privacy Manager v1.0, Q-IAS (Thales Gemalto Quantum IAS application) v1.0.0 (also called QSign).

Applet name	Package	Package AID
GDP v3.0	com.gemalto.javacardx.gdp	A00000001810020303
LDSV2 v1.1	com.gemalto.javacard.icao.lds2	A000000018300B0201000000000000FE
IAS Classic v5.2.3	com.gemalto.javacard.iasclassic	A000000018800000000066240FF
Q-IAS v1.0.0	com.thalesgroup.javacard.qsign	A0000008448000000000B4D00FF
eTravel v3.2	natif	N/A
BioPin Manager v3.1: MOC Client	com.gemalto.moc.client	4D4F43415F436C69656E74
BioPin Manager v3.1: MOC Server	com.gemalto.moc.server	4D4F43415F536572766572
MSFT PnP v1.0	com.gemalto.javacard.mspnp	A0000000308000000006DF00FF
Pure DI (version v3.05)	com.gemalto.puredi	A000000018320A0100000000000000FF
Privacy Manager v1.0	com.gemalto.javacard.eid	A0000000308000000008DB00FF
MPCOS v4.1	com.gemalto.mpcos	A000000018300301000000000000FF
FIDO Authentificator v2.1	com.gemalto.javacard.fido.ctap	A0000000308000000000A9A00FF

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

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

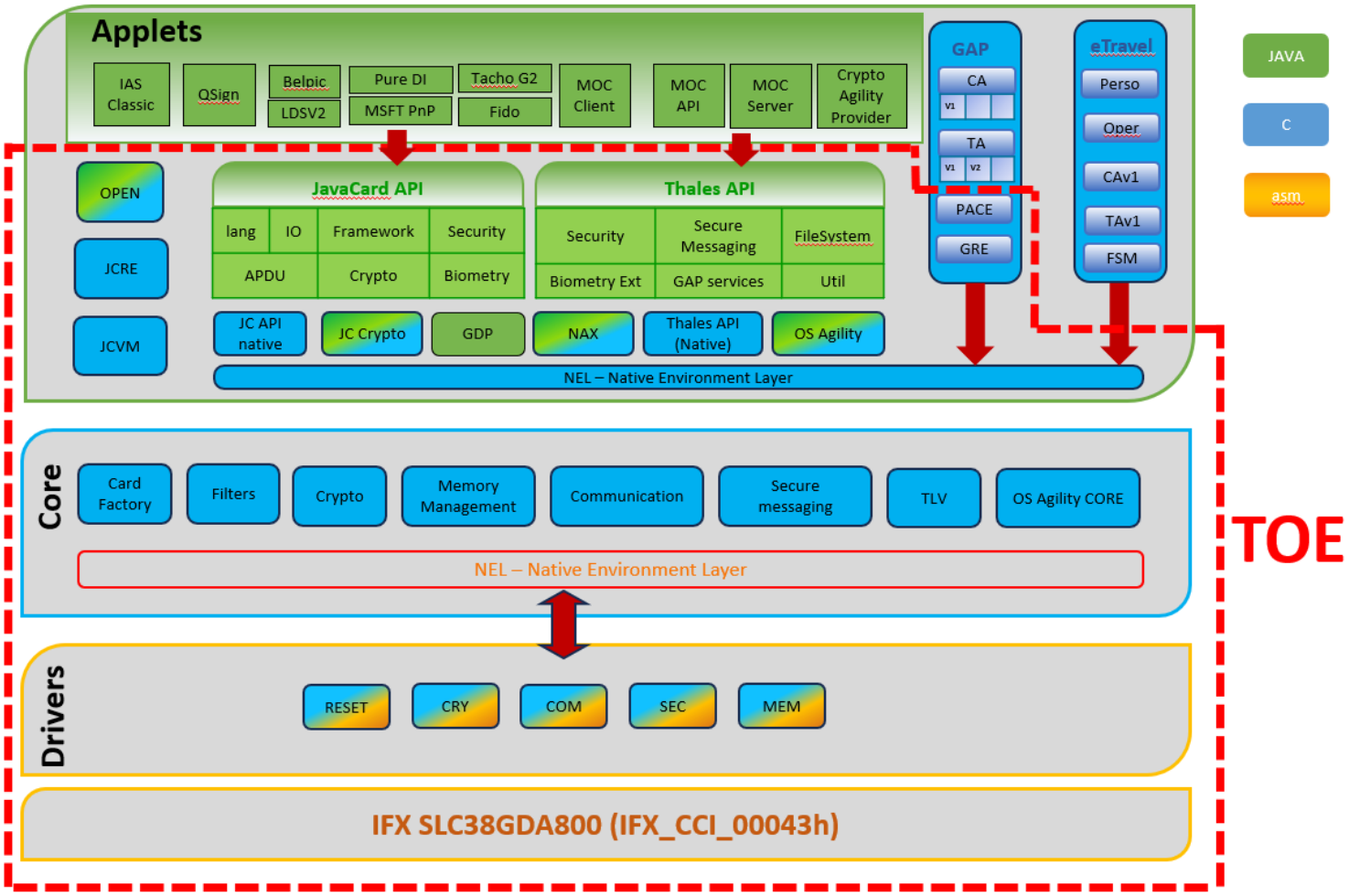


Figure 1: MultiApp V5.2 smartcard architecture

Applets and the MultiApp V5.2 Java Card platform, are located in flash code area.
All the data (related to the applets or to the Java Card platform) are located in flash data area. The separation between these data is ensured by the Java Card firewall as specified in [JCRE3].

MultiApp V5.2 products is a modular product where some features could be removed, based on the customer needs. (See identification and configuration option).

2.3 TOE BOUNDARIES

The Target of Evaluation (TOE) is the JCS open platform of the MultiApp V5.2 product. It is defined by:

- The Java Platform 3.2 based on JLEP3 Operating System
- The PACE module to provide PACE secure channel
- The underlying Integrated Circuit

Applications stored in Flash mask in code area in MultiApp V5.2, are outside the TOE.
The Applets loaded pre issuance or post issuance are outside the TOE, Other smart card product elements, (such as holograms, magnetic stripes, security printing) are outside the scope of this Security Target.

Java Card RMI is not implemented in the TOE.

2.4 TOE DESCRIPTION

2.4.1 Architecture

The MultiApp V5.2 platform is an operating system that complies with two major industry standards:

- Oracle's Java Card 3.2, which consists of the Java Card 3.2 Virtual Machine [JCVM3], the Java Card 3.2 Runtime Environment [JCRE3] and the Java Card 3.2 Application Programming Interface [JCAPI3].
- The Global Platform Card Specification version 2.3 [GP23]
- GAP: the General Authentication Procedure, for compliance with latest version of [TR03110-2]
- GAP and File System APIs: these new APIs are required for the [TR03110-2] based applications (eIDAS and new Signature application compliant to [TRSIGN]).
- GDP: Global Dispatcher Perso application to centralize application personalization (at first for eTravel).
- Support of Flash Modularity: possibility during product construction to embed only features required for a given customer item.

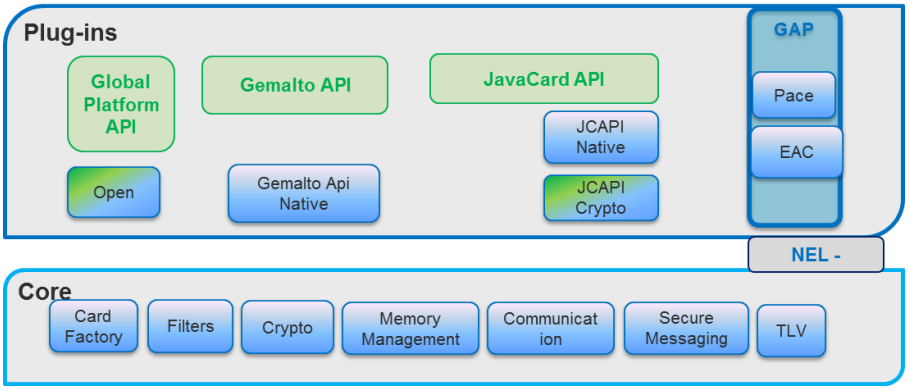


Figure 2: MultiApp V5.2 Java Card platform architecture

As described in figure 2, the MultiApp V5.2 platform contains the following components:

- **The Core layer**

  The Core layer remains unaffected as the basic smart card services (softmasks/filters, communication protocols, memory management, secure messaging) remain the same.

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

  - DES, 3DES (ECB, CBC)
  - RSA up to 4096 (CRT method & public Std method), 4096 (Std private method)
  - DH up to 3072
  - AES 128, 192, 256
  - SHA1, SHA 2 (224, 256, 384, 512), SHA3 (224, 256, 384, 512), SHAKE256
  - ECC (ECDSA et ECDH) up to 521
  - PACE DH up to 2048 Integrated Mapping, Generic Mapping
  - PACE ECDH up to 521 Integrated Mapping, Generic Mapping
  - Pseudo-Random Number Generation (PRNG) & Software random
  - Pseudonymous signature (Psign) ECC up to 521 (not evaluated)
  - CRC16, CRC32
  - HMAC SHA1, SHA 2 (up to 512)
  - ML-DSA-65 (public key 1952 bytes, private key 4032 bytes)

- **The Plug-ins layer**
  - The **Javacard Runtime Environment**

  It conforms to [JCRE3] and provides a secure framework for the execution of the Java Card programs and data access management (firewall).
  Among other features, multiple logical channels are supported, as well as extradition, DAP, Delegated management, SCP01, SCP02 and SCP03.

  - The **Javacard Virtual Machine**

  It conforms to [JCVM3]and provides the secure interpretation of bytecodes.

  - The **API**

  It includes the standard Java Card API [JCAPI3] and the Thales proprietary API.

  - The **Global Platform Issuer Security Domain**

  It conforms to [GP23] and provides card, key and applet management functions (contents and life-cycle) and security control.

  - The **GAP component**

    GAP is an extension of PACE, it provides additional commands terminal authenticate (TA) and Chip Authenticate (CA). This provides mutual authentication, secure messaging channel, authorization verified by application through specific API.

The MultiApp V5.2 platform provides the following services:

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

2.4.2 Modularity optionality concept

The MultiApp product family relies on JLEP3 OS design. The JLEP3 OS provides modularity by design. By modularity, we understand a split of the code in sub-systems, each sub-system being broken down in another set of sub-system or modules, a module being an identified list of compilation units (java, c or assembly file).
The modularity is ensured by a functional consistency of functions regrouped in the organizational units (compilation unit, module or sub-system). The modularity can be measured through the amount of dependencies across organizational units. A modular design ensures minimal dependency.
The objective is to benefit from the migration to Flash technology so that the actual generated customer item (derived from the generic product) would embed only features required and hence have an optimal memory footprint.

Flash
The solution relies on the tower concept: when building (compiling and linking) the product, the modules are combined in features. The features are assembled like bricks in towers as illustrated here:

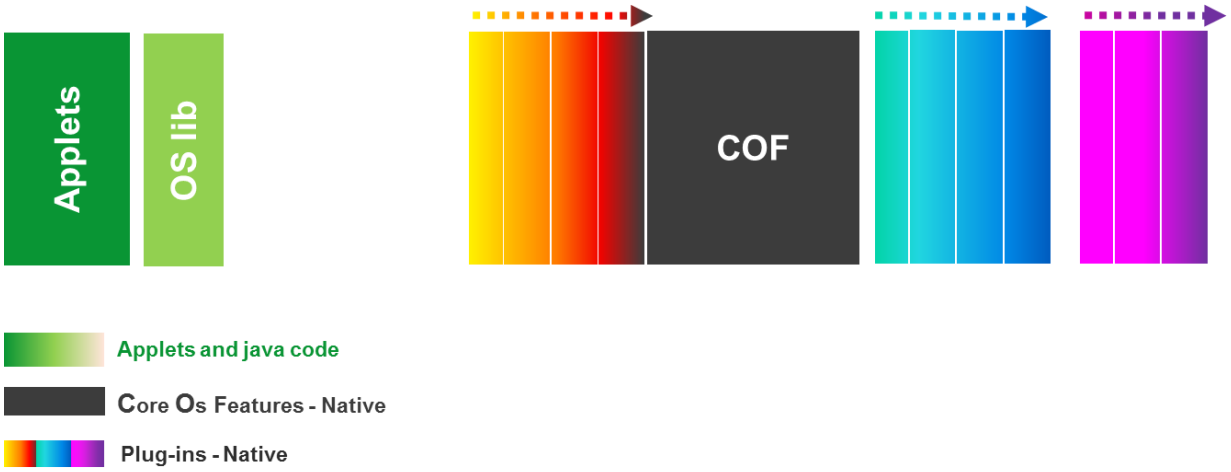


Figure 3 - Flash Modularity towers concept

Towers can be shrunk by removing features one-by-one, starting from the top of the tower (from the left on this figure), thus achieving a reduced memory footprint. It is not possible to remove a feature when the feature above it is present. This brings particular constraints to the way the features must be assembled. Typically, features in the upper layers should be the ones with no dependencies on them in order to be removable. Features in the lower layers may have dependencies only from features of layers above them. This also explains why we have core feature: they constitute the irremovable heart of the tower.

Note:
The diagram above in Figure 3 - Flash Modularity towers concept illustrates the virtual memory layout of a product release as generated by the chip manufacturer's compiler/linker tool chain. As a result, it requires managing the module/feature mapping to ensure the features are assembled as expected.

2.4.3 Agility concept

The MultiAppV5.2 product embeds an optional functionality to update the operating system when the card is already on the field. This functionality is named OS-agility.

The mechanism will allow to correct product issues and security issues when the product is already deployed. The updates are done through a dedicated application (OS-Agility Plug-in see figure 4) and are a list of instructions to update the memory.
The update instructions are packaged into a block protected in confidentiality and integrity by keys known only by Thales DIS. The block can be transmitted and executed by the card only after a successful authentication done with keys only known by the customer. Like this Thales DIS is unable to load some contents into the card without the consent of the customer and the customer also cannot load a content without the consent of Thales DIS.

The patch is transmitted to the card throw a trusted channel that can be manage by the OS agility application or thanks to the certificate update mechanism managed by the eTravel application.

Prior the execution of the instructions of the patch, some prerequisites are verified, the code ensures that the current product configuration allows the correct execution of the instructions. Some updates can be conditionally be executed following the availability of a dedicated feature (cf modularity concept in §2.4.2). At the end of the execution, the traceability elements are also updated to allow a complete identification of the product (platform version and current patch version). The patch loading mechanism ensures also the atomicity of the updates.

2.4.4 Crypto-Agility concept

This new crypto-agility concept has been introduced to deal with new algorithms availability and care about future attacks that will impact the signature algorithms present on the card. This concept allows products to be updated in the field without recalling them and redeploying new ones. Crypto-agility is based on OS-agility feature (reviewed in §2.4.3) that already certified in similar previous products and also based on GDP application that allows to re-personalize application with new Signature algorithm and new Signature key.

New PQC algorithms do not have all the maturity that the others already used for years may have, this is why crypto-agility will allow to go further and allow to add/remove and activate/deactivate a PQC signature algorithm.
For example, crypto-agility can change the length of the key in the field without impacting the code of the applications using this key.

2.4.5 Architecture: design view by features

It is important to distinguish the functional design view of the platform, described in chapter 2.4.1, from the representation of the products features. Design sub-systems and features have a common definition: they are a collection of modules. Sub-systems are a design group, while features are functional groups of modules.

The following diagram shows a high level representation of the MultiApp architecture by feature:

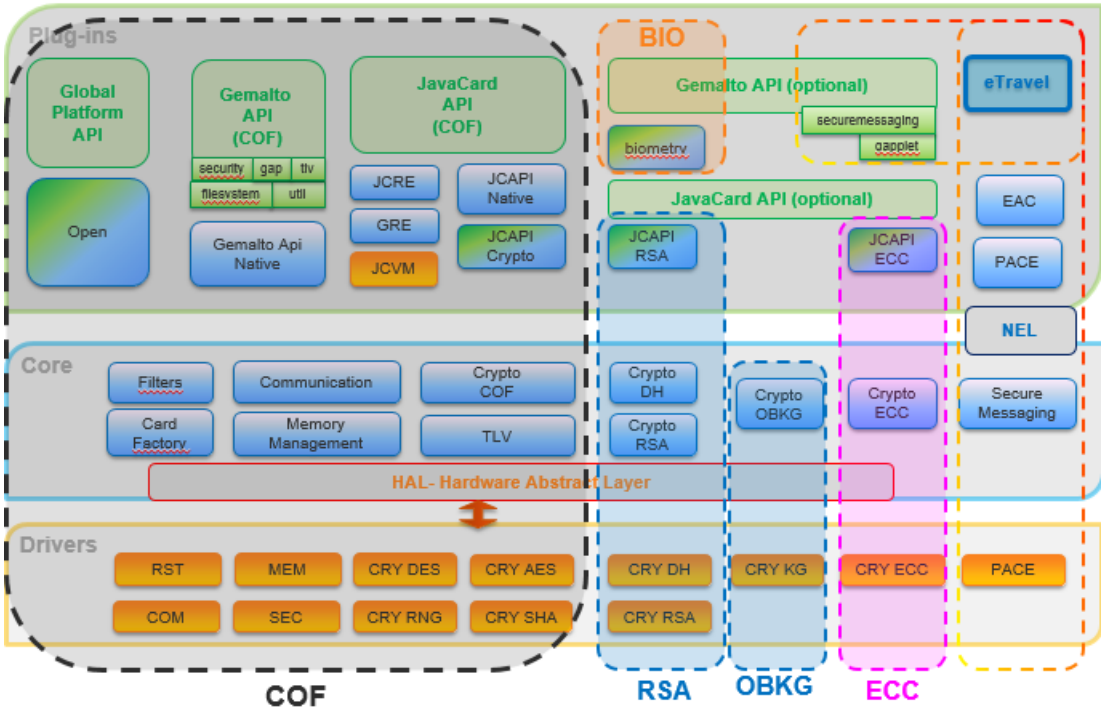


Figure 4: MultiApp design by features

Note that the COF (Core Operating Feature) shows all the mandatory features, other elements are consider as additional bricks. These bricks could be removed.

2.5 LIFE-CYCLE

2.5.1 Product Life-cycle

2.5.1.1 Actors

Actors	Identification
Integrated Circuit (IC) Developer	Infineon
Embedded Software Developer (Also named OS developer for the phase 1 of the Life cycle)	Thales DIS
Integrated Circuit (IC) Manufacturer	Infineon
Module Manufacturer	Thales DIS or Infineon
Form factor Manufacturer (optional)	Thales DIS or other (when it is done before the TOE delivery) It can be also a third party company or the SC Issuer after the TOE delivery
Card manufacturer (Initializer/Pre-personalizer)	THALES It can be also a third party company or the SC Issuer after the TOE delivery (especially for the Wafer Init process)

Actors	Identification
Personalization Agent (Personalizer)	The agent who is acting on the behalf of the Issuer (e.g. issuing State or Organization) and personalize the TOE and applicative data (e.g. MRTD for the holder) by activities establishing the identity of the user (e.g. holder with biographic data).
OS Update loader	Agent who is acting on the behalf of the issuer to load the OS patch on the card
Issuer	The Issuer is the actual owner of the SE. As such, no OS Update operation shall be made without his consent. This concept has already been introduced in the SE PP.
Card Holder	The rightful holder of the card for whom the issuer personalizes it.

Table 2: Identification of the actors

2.5.1.2 *Life cycle description*

For this product, wafer init process shall be ignored, only Classic init process shall be considered.

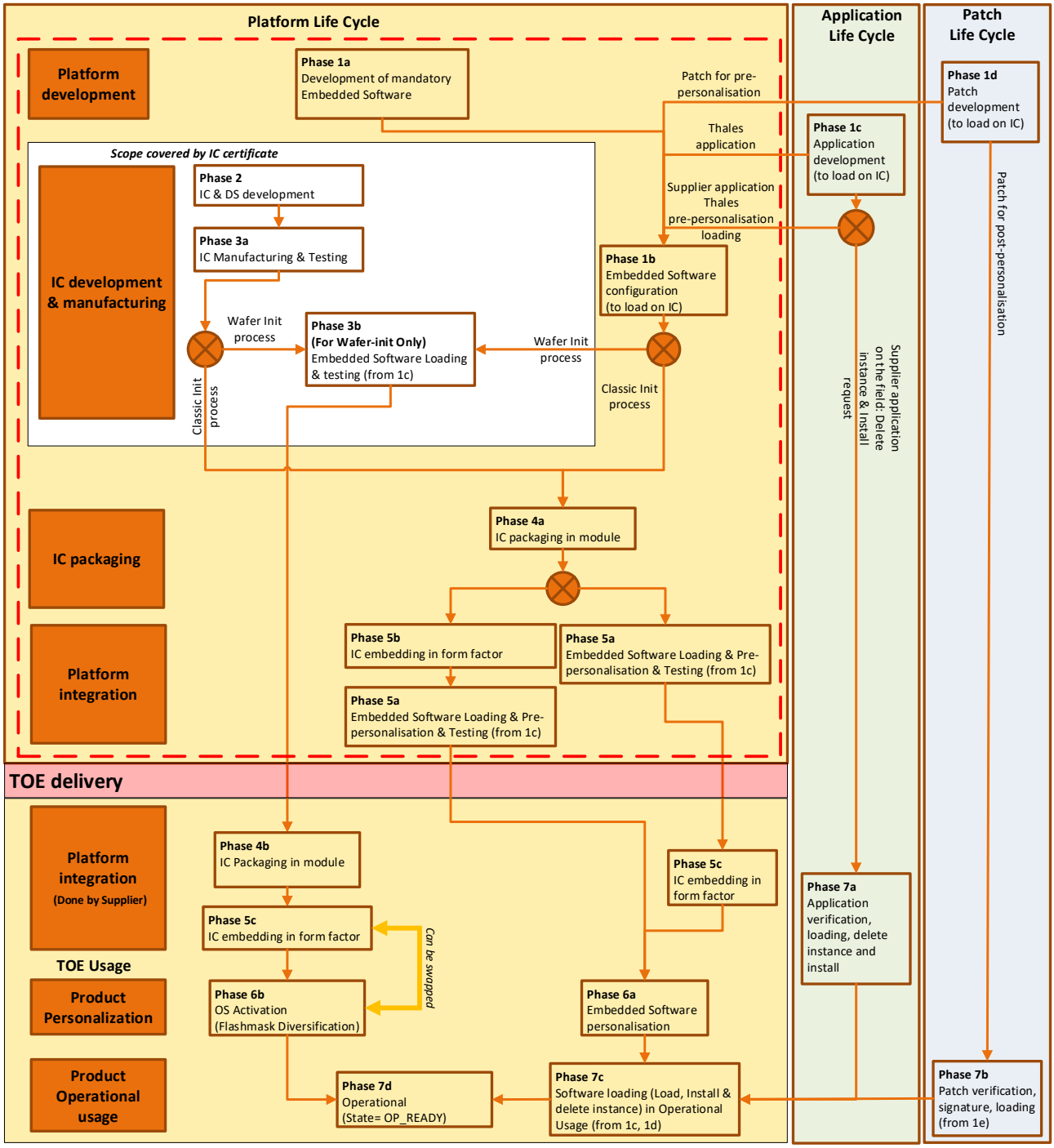


Figure 5: Manufacturing phases description

The Life cycle is described on the figure hereunder:

Phase	Description / comments		Who	Where
1	MAV5.2 platform development	Platform development & tests (1.a)	**Thales** R&D team SL Crypto team - secure environment -	Thales Development site (see §2.5.4)
	Thales applets (IAS, eTravel…) development	- Applet Development (1.c) - Applet tests	**Thales** R&D team - secure environment -	Thales Development site (see §2.5.4)
	Patch development	- Patch Development (1.d) - Patch tests	**Thales** R&D team - secure environment -	Thales Development site (see §2.5.4)
	PSE team	- Platform configuration (1.b) - Script development	**Thales Thales Product Engineering**	Thales manufacturing site (see §2.5.4)
2	IC development	IFX_CCI_000043h development	**Infineon** - Secure environment -	Infineon development site(s)
3a 3b (Optional)	IC manufacturing & Testing Initialization / Pre-personalization	Manufacturing of virgin IFX_CCI_000043h integrated circuits embedding the Thales Design Services flash loader, and protected by a dedicated transport key. Loading of the Thales software (platform and applets on top based on script generated) – For WaferInit process only	**Infineon** - Secure environment -	Infineon development site(s)
4a	SC manufacturing: IC packaging & Embedding, also called “assembly”	- IC packaging & testing	**Thales** Production teams - Secure environment – Third party company	Thales manufacturing site (see §2.5.4) Third party company site
5.a	Initialization / Pre-personalization (Not Applicable for wafer-init process)	Loading of the Thales software (platform and applets on top based on script generated)	**Thales** Production teams - Secure environment -	Thales manufacturing site (see §2.5.4)
5.b	Embedding	Put the module on a dedicated form factor (Card, inlay MFF2, other…)	**Thales** Production teams - Secure environment – Third party company	Thales manufacturing site (see §2.5.4) Third party company site
4b	IC packaging & Embedding, also called “assembly” (Wafer Init)	- IC packaging & testing	SC Issuer or another Third party company	SC Personalizer or Third party company site
5c	Embedding	Put the module on a dedicated form factor (Card, inlay MFF2, other…)	SC Issuer or another Third party company	SC Personalizer or Third party company site
6a	SC Personalization	Creation of files and loading of end-user data	SC Issuer or Another Third party company	SC Personalizer or Third party company site
6b	OS Activation (Wafer Init)	Launch the card activation process (Flashmask key diversification) following by the Personalisation (Creation of files and loading of the end-user data)	SC Issuer or Third party company	SC Personalizer or Third party company site
7	End-usage	Application verification before loading (7.a)	SC Issuer	Field
		Application Loading (Load, Install and delete instance capabilities) (7.c)	SC Issuer	Field
		Patch verification before loading (Signature) (7.b)	Thales	Field
		Patch update (7.b)	Thales	Field
		End-usage for cardholder (7.d)	Cardholder	Field

Figure 6: Life Cycle description

Remark 1: Initialization & pre-personalization operation could be done on module or on other form factor. The form factor does not affect the TOE security.

Remark 2: For initialization/pre-personalization IC flash loader could be used based on the IC manufacturer recommendation.

Remark 3: Embedding (module put on a dedicated form factor) will be done on an audited site if the *Embedding phase (5a)* is before the TOE delivery.

Remark 4: for step 4a, if the module is contact less only, Infineon is then the third party company. If the module is combi or contact, Thales is the third party company.

Remark 5: for step 5b, the smartcard is protected by mutual authentication and third party company which is responsible of the inlay process, is considered out of evaluation. Please refer to the ANSSI NOTE/09.1 §2.3.

2.5.2 TOE Life-cycle

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

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

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

The JCS storage is not necessarily a single step in the life cycle since it can be stored in parts. The JCS delivery occurs before storage and may take place more than once if the TOE is delivered in parts.
These four stages map to the product life cycle phases as shown in Figure 6.

As a summary description of how the parts of the TOE are delivered to the final customer, the MultiApp V5.2 application is delivered mainly in form of a smart card or inlay. The form factor is packaged on Thales DIS's manufacturing facility and sent to final customer premises.

The different guides accompanying the TOE and parts of the TOE are the ones specified in [AGD] section. They are delivered in form of electronic documents (*.pdf) by Thales DIS's Technical representative.

Note related to patch development

No patch is present within the TOE for the present evaluation. Indeed, should a patch be needed in the future, it would require at least a maintenance of the CC certificate, as required by the CC scheme rules. However, the patch mechanism is part of the TOE and as such its security is assessed within the present evaluation.

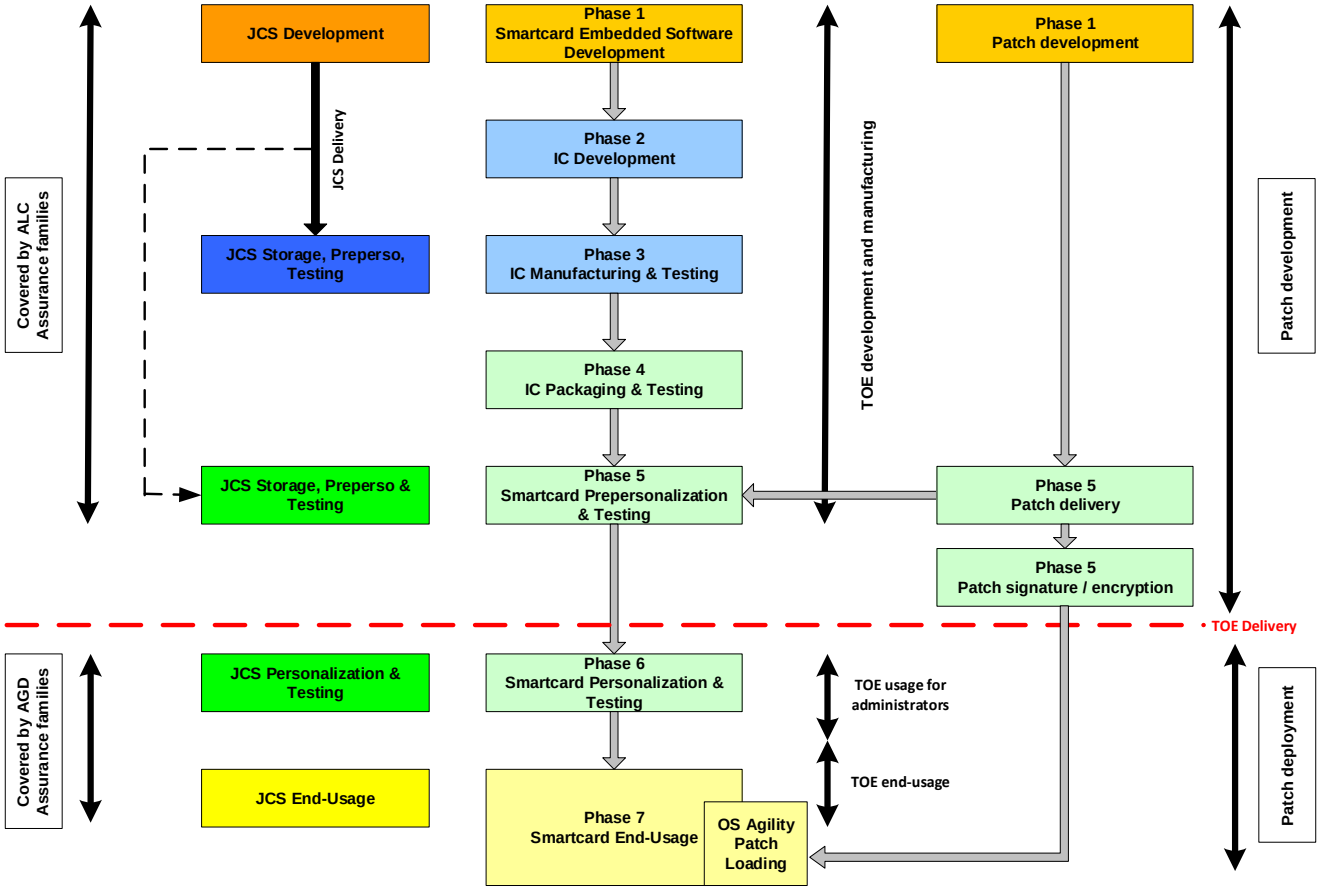


Figure 7: JCS (TOE) Life Cycle within Product Life Cycle

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

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

In case of wafer-init process, the Pre-Personalizer (Thales Design Services in this case) may store, pre-personalize the JCS and potentially conduct tests on behalf of the JCS developer. The SC Pre-Personalization environment shall protect the integrity and confidentiality of the JCS and of any related material.

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

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

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

The JCS final usage environment is that of the product where the JCS is embedded in. It covers a wide spectrum of situations that cannot be covered by evaluations. The JCS and the product shall provide the full set of security functionalities to avoid abuse of the product by untrusted entities.
Note: Potential applications loaded in pre-issuance will be verified using dedicated evaluated verification process. Applications loaded in post-issuance will need to follow dedicated development rules.

2.5.3 GP Life-cycle

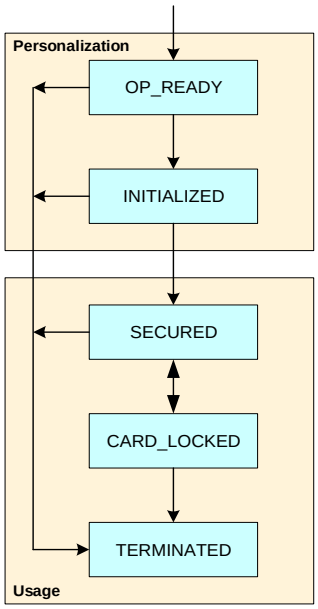


Figure 8: GP Life Cycle

Note that the Patch management (OS-Agility mechanisms) will be available only for the mode:

- OP_READY
- INITIALIZED
- SECURED

2.5.4 Involved Thales-DIS sites

- **Development and Project Managment**
  - La Ciotat (France), Meudon (France), Vantaa, Singapore
    - CC project management Platform development & eTravel development & support
- **Manufacturing**
  - Gémenos, Singapore, Vantaa, Tczew, Curitiba, Chanhassen, Pont-Audemer, Montgomeryville
- **IT activities**
  - Gémenos, Calamba, Chennai, Noida, Paris (TELEHOUSE), Elancourt

2.6 TOE INTENDED USAGE

2.6.1.1 Personalization Phase

During the Personalization Phase the following Administrative Services are available:

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

If the OS Agility is available

- Patch Management

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

2.6.1.2 Usage Phase

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

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

In addition, the following User services are available:

- Applet Selection
- Applet Interface

If the OS Agility is available

- Patch Management

2.6.1.3 NON-TOE HARDWARE/SOFTWARE/FIRMWARE REQUIRED BY THE TOE

In order to manage distant secure channel according to [GP23], a remote system must be able to establish a connection with TOE and therefore must possess shared secret with TOE.

Applets are supposed to be used with the platform to communicate to external world. Applet can create a dedicated secure channel using platform services. In such case, a remote system must be able to establish a connection with applet and therefore must possess shared secret with applet.

In order to manage local PACE secure channel, only local terminals possessing authorization information (a shared secret stored or retrieved by terminal (as PIN, CAN or MRZ) or secret derived from shared secret) can get access to the user data stored on the TOE and use security functionality.

Application note: Definition of local terminal is a refinement from the one in [PP_EAC2] but without direct reference to travel document allowing usage of PACE secure channel for several purposes including travel document but not exclusively.

2.6.1.4 TOE Delivery

As a summary description of how the parts of the TOE are delivered to the final customer, the MultiApp V5.2 embedded software is delivered mainly in form of a smart card, module or wafer. The form factor is packaged on Thales's manufacturing facility and sent to final customer premises or via the wafer init process from the IC Manufacturer premises.

The product is sent to the customer by standard transportation respecting Thales Transport Security Policies.

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

Item type	Item	Reference/Version	Form of delivery
Software and Hardware	MultiApp V5.2	Refer to paragraph §1.3	Smart card, module or wafer
Document	MultiApp V5.2: AGD_OPE document - Javacard Platform	D1600884 V1.8	Electronic document via secure file download
Document	MultiApp V5.2: AGD_PRE document - Javacard Platform	D1600885 V1.8	Electronic document via secure file download
Document	MultiApp ID V5 Operating System Reference Manual	D1525385D December 20, 2023	Electronic document via secure file download
Document	MultiApp Guidance Document - Guidance document for secure development for MultiApp products	D1539156 V1.3.A.1	Electronic document via secure file download
Document	Guide for CC certified PQC Signatures	D1610996 V1.5	Electronic document via secure file download

3 CONFORMANCE CLAIMS

3.1 CC CONFORMANCE CLAIM

Common criteria Version:

This ST conforms to CC Version 3.1 revision 5 [CC-1] [CC-2] [CC-3].

Conformance to CC part 2 and 3:

- CC part 2 extended with the FCS_RNG, FMT_LIM.1, FMT_LIM.2 and FPT_EMS.1 components. All the other security requirements have been drawn from the catalogue of requirements in Part 2 [CC-2].
- CC part 3 conformant.

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

3.2 PP CLAIM

The MultiApp V5.2 JCS security target claims strict conformance to the Protection Profile “JavaCard System – Open configuration”, ([PP-JCS-Open]).

The MultiApp V5.2 JCS security target is a composite security target, including the IC security target [IFX-IC]. However, the security problem definition, the objectives, and the SFR of the IC are not described in this document.

3.3 PACKAGE CLAIM

This ST is conforming to assurance package EAL6+ augmented with ALC_FLR.1 defined in CC part 3 [CC-3].

3.4 CONFORMANCE STATEMENT

This ST strictly conforms to [PP-JCS-Open]. The conformance is explained in the rationale.
Items relative to PACE module from [PP_EAC2] have been added to perform composite evaluation but no conformance to [PP_EAC2] is required.

4 SECURITY ASPECTS

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

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

TSFs must be continuously active in one way or another; this is called “OPERATE”.

4.1 CONFIDENTIALITY

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

4.2 INTEGRITY

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

4.3 UNAUTHORIZED EXECUTIONS

#.EXE-APPLI-CODE	Application (byte) code must be protected against unauthorized execution. This concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language ([JAVASPEC]§6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code.;
#.EXE-JCS-CODE	Java Card System bytecode must be protected against unauthorized execution. Java Card System bytecode includes any code of the Java Card RE or API. This concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language ([JAVASPEC]§6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code. Note that execute access to native code of the Java Card System and applications is the concern of #.NATIVE.
#.FIREWALL	The Firewall shall ensure controlled sharing of class instances, and isolation of their data and code between CAP files (that is, controlled execution contexts) as well as between CAP files and the JCRE context. An applet shall neither read, write nor compare a piece of data belonging to an applet that is not in the same context, nor execute one of the methods of an applet in another context without its authorization.
#.NATIVE	Because the execution of native code is outside of the JCS TSF scope, it must be secured so as to not provide ways to bypass the TSFs of the JCS. Loading of native code, which is as well outside the TSFs, is submitted to the same requirements. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

4.4 BYTECODE VERIFICATION

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

4.4.1 CAP file Verification

Bytecode verification includes checking at least the following properties: (1) bytecode instructions represent a legal set of instructions used on the Java Card platform; (2) adequacy of bytecode operands to bytecode semantics; (3) absence of operand stack overflow/underflow; (4) control flow confinement to the current method (that is, no control jumps to outside the method); (5) absence of illegal data conversion and reference forging; (6) enforcement of the private/public access modifiers for class and class members; (7) validity of any kind of reference used in the bytecodes (that is, any pointer to a bytecode, class, method, object, local variable, etc actually points to the beginning of piece of data of the expected kind); (8) enforcement of rules for binary compatibility (full details are given in [JCVM3], [JVM], [JCBV]). The actual set of checks performed by the verifier is implementation-dependent, but shall at least enforce all the “must clauses” imposed in [JCVM3] 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 CAP file required by the loaded applet is indeed on the card, in a binary-compatible version (binary compatibility is explained in [JCVM3] §4.4), second, that the export files used to check and link the loaded applet have the corresponding correct counterpart on the card.

4.4.2 Integrity and Authentication

Verification off-card is useless if the application CAP files 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 CAP file verification and CAP file installation.

Once a verification authority has verified the CAP file, it signs it and sends it to the card. Prior to the installation of the CAP file, the card verifies the signature of the CAP file, which authenticates the fact that it has been successfully verified. In addition to this, a secured communication channel is used to communicate it to the card, ensuring that no modification has been performed on it.

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

4.4.3 Linking and Verification

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

4.5 CARD MANAGEMENT

#.CARD-MANAGEMENT	(1) The card manager (CM) shall control the access to card management functions such as the installation, update or deletion of applets. (2) The card manager shall implement the card issuer's policy on the card.
#.INSTALL	(1) The TOE must be able to return to a safe and consistent state should the installation of a CAP file or an applet fail or be cancelled (whatever the reasons). (2) Installing an applet must have no effect on the code and data of already installed applets. The installation procedure should not be used to bypass the TSFs. In short, it is an atomic operation, free of harmful effects on the state of the other applets. (3) The procedure of loading and installing a CAP file shall ensure its integrity and authenticity. In case of Extended CAP files, installation of a CAP shall ensure installation of all the packages in the CAP file.
#.SID	(1) Users and subjects of the TOE must be identified. (2) The identity of sensitive users and subjects associated with administrative and privileged roles must be particularly protected; this concerns the Java Card RE, the applets registered on the card, and especially the default applet and the currently selected applet (and all other active applets in Java Card System 2.2). A change of identity, especially standing for an administrative role (like an applet impersonating the Java Card RE), is a severe violation of the Security Functional Requirements (SFR). Selection controls the access to any data exchange between the TOE and the CAD and therefore, must be protected as well. The loading of a CAP file 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 CAP files) 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. CAP file deletion shall make the code of the CAP file no longer available for execution. In case of Extended CAP files, deletion of a CAP shall ensure that code and data for all the packages in the CAP file is 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 CAP file are functionally different operations and may obey different security rules. For instance, specific CAP files can be declared to be undeletable (for instance, the Java Card API CAP files), or the dependency between installed CAP files may forbid the deletion (like a CAP file using super classes or super interfaces declared in another CAP file).

4.6 SERVICES

#.ALARM	The TOE shall provide appropriate feedback upon detection of a potential security violation. This particularly concerns the type errors detected by the bytecode verifier, the security exceptions thrown by the Java Card VM, or any other security-related event occurring during the execution of a TSF.
#.OPERATE	(1) The TOE must ensure continued correct operation of its security functions. (2) In case of failure during its operation, the TOE must also return to a well-defined valid state before the next service request.
#.RESOURCES	The TOE controls the availability of resources for the applications and enforces quotas and limitations in order to prevent unauthorized denial of service or malfunction of the TSFs. This concerns both execution (dynamic memory allocation) and installation (static memory allocation) of applications and CAP files.
#.CIPHER	The TOE shall provide a means to the applications for ciphering sensitive data, for instance, through a programming interface to low-level, highly secure cryptographic services. In particular, those services must support cryptographic algorithms consistent with cryptographic usage policies and standards.
#.KEY-MNGT	The TOE shall provide a means to securely manage cryptographic keys. This includes: (1) Keys shall be generated in accordance with specified cryptographic key generation algorithms and specified cryptographic key sizes, (2) Keys must be distributed in accordance with specified cryptographic key distribution methods, (3) Keys must be initialized before being used, (4) Keys shall be destroyed in accordance with specified cryptographic key destruction methods.
#.PIN-MNGT	The TOE shall provide a means to securely manage PIN objects. This includes: (1) Atomic update of PIN value and try counter, (2) No rollback on the PIN-checking function, (3) Keeping the PIN value (once initialized) secret (for

	instance, no clear-PIN-reading function), (4) Enhanced protection of PIN's security attributes (state, try counter…) in confidentiality and integrity.
#.SCP	The smart card platform must be secure with respect to the SFRs. Then: (1) After a power loss, RF signal loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state. (2) It does not allow the SFRs to be bypassed or altered and does not allow access to other low-level functions than those made available by the CAP files of the Java Card API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System. (3) It provides secure low-level cryptographic processing to the Java Card System. (4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism. (5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low–level control accesses (segmentation fault detection). (6) It safely transmits low–level exceptions to the TOE (arithmetic exceptions, checksum errors), when applicable. Finally, it is required that (7) the IC is designed in accordance with a well defined set of policies and standards (for instance, those specified in [PP-IC-0035]), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.
#.TRANSACTION	The TOE must provide a means to execute a set of operations atomically. This mechanism must not jeopardise the execution of the user applications. The transaction status at the beginning of an applet session must be closed (no pending updates).

5 SECURITY PROBLEM DEFINITION

5.1 ASSETS

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

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

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

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

5.1.1 User data

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

D.APP_C_DATA
Confidential sensitive data of the applications, like the data contained in an object, a static field, 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 and the PIN security attributes (PIN Try limit, PIN Try counter and State). To be protected from unauthorized modification.

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

D.PIN
Any end-user's PIN.
To be protected from unauthorized disclosure and modification.

5.1.2 TSF data

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

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

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

D.JCS_DATA

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

D.SEC_DATA

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

5.1.3 Supplementary assets

The following assets are related to patch management in post-issuance phase (phase 7). As mentioned in section 2.5.2, there is no patch associated to the present TOE, however the patch mechanisms are within the evaluation scope.

D.OS-UPDATE_DEC-KEY	Refinement of D.APP_KEYS. It is a Thales DIS cryptographic key (K_{ENC}), owned by the OS Developer, and used by the TOE to decrypt the additional code to be loaded. Note: no assumption is made on the type of this decryption key, i.e. it can be either a symmetric key or the secret component of an asymmetric key pair. To be protected from unauthorized disclosure and modification.
D.OS-UPDATE_SGNVER-KEY	Thales DIS keys used for the signature It is a Thales DIS cryptographic key (K_{MAC}), owned by the OS Developer, and used by the TOE to verify the signature of the additional code to be loaded. Note: no assumption is made on the type of this signature verification key, i.e. it can be either a symmetric key or the public component of an asymmetric key pair. Case of a symmetric key: to be protected from unauthorized disclosure and modification. Case of an asymmetric public key: to be protected from unauthorized modification.
D.OS-UPDATE_ADDITIONALCODE	Code to be added to the OS after TOE issuance. The additional code has to be signed by the OS Developer. After successful verification of the signature by the Initial TOE, the additional code is loaded and installed/activated through an atomic activation (to create an Updated TOE). To be protected from unauthorized disclosure and modification.
D.OS-UPDATE-CODE-ID	Identification data associated to the additional code. It is loaded and/or updated in the same atomic operation as additional code loading. To be protected from unauthorized modification. Application Note: The identification data (D.OS-UPDATE-CODE-ID) may be also protected from unauthorized disclosure (confidentiality requirement) to not permitting an attacker to determine if a given TOE has been updated or not (even if it is not possible to distinguish between functional and security updates). However, confidentiality is not mandatory since in most cases the identification data must be readily available on the field through technical commands, even in the TERMINATED state.

5.2 ITEMS FOR PACE MODULE

Application note: Definition of asset associated to PACE module is a refinement from the one in [PP_EAC2] but without direct reference to travel document allowing usage of PACE secure channel for several purposes including travel document but not exclusively.

5.2.1 Primary assets or user data

Object No.	Asset	Definition	Generic security property to be maintained by the current security policy
1	user data stored on the TOE (requiring PACE secure channel)	All data (being not authentication data) being allowed to be read out solely by an authenticated terminal acting as Basic Inspection System with PACE (in the sense of [ICAO-TR-SAC]).	Confidentiality Integrity Authenticity
2	user data transferred between the TOE and the terminal connected (i.e. an authority represented by Basic Inspection System with PACE)	All data (being not authentication data) being transferred between the TOE and an authenticated terminal acting as Basic Inspection System with PACE (in the sense of [ICAO-TR-SAC]). User data can be received and sent (exchange ⇔ {receive, send}).	Confidentiality Integrity Authenticity

Table 3: Primary Assets

Note: Unavailability in a sense of non-disclosure of data allowing user traceability.

5.2.2 Secondary assets and TSF data

The secondary assets also having to be protected by the TOE in order to achieve a sufficient protection of the primary assets are listed in the following table. The secondary assets represent TSF and TSF-data in the sense of the CC.

Object No.	Asset	Definition	Generic security property to be maintained by the current security policy
4	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 authorised subjects only.	Availability
5	PACE establishment authorization data	Restricted-revealable authorization information for a human user being used for verification of the authorization attempts as authorized user (PACE password). These data are stored in the TOE and are not to be send to it.	Confidentiality Integrity
6	TOE internal secret cryptographic keys	Permanently or temporarily stored secret cryptographic material used by the TOE in order to enforce its security functionality.	Confidentiality Integrity
7	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.	Integrity Authenticity

Table 4: Secondary Assets

Note: PACE passwords are not to be sent to the TOE.

5.2.3 Subjects and external entities

The ST considers the following external entities and subjects for PACE usage:

External Entity No.	Role	Definition
1	Application user (e.g. travel document holder).	This entity is commensurate with application user for whom the Issuer has personalised the PACE part of the TOE and therefore may use PACE secure channel (e.g. 'MRTD Holder' in [PP-BAC])
2	Application user (e.g. travel document presenter)	This entity is commensurate with application user with usage of PACE secure channel to be authenticated (e.g. 'Traveller' in [PP-BAC])
3	Terminal	A terminal is any technical system communicating with the TOE through the contactless/contact interface and being recognised by the TOE as not being PACE authenticated. This entity is commensurate with 'Terminal' in [PP-BAC].
4	PACE Terminal (e.g. Basic Inspection System with PACE (BIS-PACE)	A local system communicating with the TOE and implementing the terminal's part of the PACE protocol. This entity is commensurate with BIS-PACE in [PP-PACE].
5	Personalisation Agent	This entity is commensurate with 'Personalisation agent' in [PP-BAC].
6	Manufacturer	This entity is commensurate with 'IC Manufacturer' and FF Manufacturer and Pre-personalizer roles as defined in §2.5.1.2 Life cycle description.
7	Attacker	This external entity is commensurate with 'Attacker' in [PP-BAC].

Table 5: Subjects and External Entities

5.3 THREATS FROM JAVA CARD SYSTEM PROTECTION PROFILE – OPEN CONFIGURATION

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

5.3.1 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, ARRAY_VIEWS_CONF, D.PIN,** and **D.APP_KEYs**.

T.CONFID-JCS-CODE

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

T.CONFID-JCS-DATA

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

5.3.2 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 CAP file 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, ARRAY_VIEWS_INT, D.PIN,** and **D.APP_KEYs**.

T.INTEG-APPLI-DATA.LOAD

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

T.INTEG-JCS-CODE

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

T.INTEG-JCS-DATA

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

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

5.3.3 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 and D.APP_KEYs.

T.SID.2

The attacker modifies the TOE's attribution of a privileged role (e.g. default applet and currently selected applet), which allows illegal impersonation of this role. See #.SID for further details.
Directly threatened asset(s): D.SEC_DATA (any other asset may be jeopardized should this attack succeed, depending on whose identity was forged).

5.3.4 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 security function such as the firewall. See #.NATIVE for details.
Directly threatened asset(s): **D.JCS_DATA**.

5.3.5 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**.

5.3.6 Card management

T.DELETION

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

T.INSTALL

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

5.3.7 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**.

5.3.8 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.
This threat refers to the point (7) of the security aspect #.SCP, and all aspects related to confidentiality and integrity of code and data.

5.4 THREATS ASSOCIATED TO PACE MODULE

Application note: Threats in this paragraph are refined form [PP_EAC2] in a more generic form in order to be applicable to any application requiring PACE protocol and not only MTRD.

T.Skimming Capturing Card-Terminal Communication

Adverse action: An attacker imitates a PACE terminal (e.g. inspection system) in order to get access to the user data stored on or transferred between the TOE and the use (e.g. 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 application data (e.g. logical travel document data).

Application Note 11: MRZ is printed and CAN is printed or stuck on the travel document.
Please note that neither CAN nor MRZ effectively represent secrets, but are restricted-revealable, cf. OE.User_Obligations.

T.Eavesdropping Eavesdropping on the communication between the TOE and the PACE terminal

Adverse action: An attacker is listening to the communication between the TOE (e.g. travel document) and the PACE authenticated terminal (e.g. 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 application data (e.g. logical travel document data).

T.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 initialization and personalization in the operational phase after delivery to the Application user*.

Threat agent: having high attack potential, being in possession of one or more legitimate application data requiring PACE usage (e.g. travel document for MRTD).

Asset: integrity and authenticity of the application data requiring PACE usage (e.g. travel document for MRTD), availability of the functionality for the application data requiring PACE usage (e.g. travel document for MRTD).

Application note: for MRTD, Application user* is travel document holder

T.Information_Leakage Information Leakage from travel 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 TOE and associated applications (e.g. travel 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 including associated applications data requiring PACE usage (e.g. travel document for MRTD).

T.Phys-Tamper Physical Tampering

Adverse action: An attacker may perform physical probing of the TOE and associated applications (e.g. travel 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 TOE and associated applications (e.g. travel 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 TOE and associated application data (e.g. travel document).

Threat agent: high attack potential, being in possession of one or more legitimate TOE and associated applications (e.g. travel documents).

Asset: integrity and authenticity of the TOE and associated application data (e.g. travel document), availability of the functionality of the TOE and associated application data (e.g. travel document), confidentiality of User Data and TSF-data of the TOE and associated application data (e.g. travel document)

T.Malfunction Malfunction due to Environmental Stress

Adverse action: An attacker may cause a malfunction of the TOE (hardware and software) and associated applications 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 TOE and associated applications (e.g. travel document) outside the normal operating conditions, exploiting errors in the TOE and associated applications (e.g. travel document) 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 TOE and associated applications (e.g. travel documents), having information about the functional operation

Asset: integrity and authenticity of the TOE and associated applications (e.g. travel document), availability of the functionality of the TOE and associated applications (e.g. travel document), confidentiality of User Data and TSF-data of the TOE and associated applications (e.g. travel document).

Application note: A malfunction of the TOE may also be caused using a direct interaction with elements on the chip surface. This is considered as being a manipulation (refer to the threat T.Phys-Tamper) assuming a detailed knowledge about TOE's internals.

T.Forgery Forgery of Data

Adverse action: An attacker fraudulently alters the User Data or/and TSF-data stored on TOE or associated application (e.g. the travel document) or/and exchanged between the TOE and the terminal connected in order to outsmart the PACE authenticated terminal (e.g. BIS-PACE by means of changed Application user 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 travel document

Application note: Application user data is travel document holder data for MRTD (e.g. biographic or biometric data)

5.5 SUPPLEMENTARY THREATS

The following threats are related to patch loading in post-issuance.

T.UNAUTHORIZED_TOE_CODE_UPDATE

An attacker attempts to update the TOE code with a malicious update that may compromise the security features of the TOE.
Targeted asset(s): D.OS-UPDATE_ADDITIONALCODE, D.JCS_CODE, D.JCS_DATA .

T.FAKE-SGNVER-KEY

An attacker modifies the signature verification key used by the TOE to verify the signature of the additional code. Hence, he is able to sign and successfully load malicious additional code inside the TOE.
Targeted assets: D.OS-UPDATE_SGNVER-KEY, D.OS-UPDATE_ADDITIONALCODE.

T.WRONG-UPDATE-STATE

An attacker prevents the OS Update operation to be performed atomically, resulting in an inconsistency between the resulting TOE code and the identification data:

- The additional code is not loaded within the TOE, but the identification data is updated to mention that the additional code is present;
- The additional code is loaded within the TOE, but the identification data is not updated to indicate the change.

Targeted asset: D.OS-UPDATE-CODE-ID.

T.INTEG-OS-UPDATE_LOAD

The attacker modifies (part of) the additional code when it is transmitted to the TOE for installation.
Targeted assets: D.OS-UPDATE_ADDITIONALCODE, D.JCS_CODE, D.JCS_DATA.

T.CONFID-OS-UPDATE_LOAD

The attacker discloses (part of) the additional code when it is transmitted to the TOE for installation.
Targeted assets: D.OS-UPDATE_ADDITIONALCODE, D.JCS_CODE, D.JCS_DATA.

5.6 ORGANIZATIONAL SECURITY POLICIES

5.6.1 OSP From Java Card System Protection Profile – Open Configuration

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

OSP.VERIFICATION

This policy shall ensure the consistency between the export files used in the verification and those used for installing the verified file. The policy must also ensure that no modification of the file is performed in between its verification and the signing by the verification authority. See #.VERIFICATION for details.
If the application development guidance provided by the platform developer contains recommendations related to the isolation property of the platform, this policy shall also ensure that the verification authority checks that these recommendations are applied in the application code.

5.6.2 TOE additional OSP

OSP.SpecificAPI

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

OSP.RNG

This policy shall ensure the entropy of the random numbers provided by the TOE to applet using [JCAPI3] is sufficient. Thus attacker is not able to predict or obtain information on generated numbers.

OSP to manage patch loading

OSP.ATOMIC_ACTIVATION

Additional code has to be loaded and installed on the Initial TOE through an atomic activation to create the Updated TOE.
Each additional code shall be identified with unique Identification Data. During such atomic activation, identification Data of the Initial TOE have to be updated to clearly identify the Updated TOE.
In case of interruption or incident during activation, the TOE shall remain in its initial state or fail secure.

OSP.TOE_IDENTIFICATION

Identification Data of the resulting Updated TOE shall identify the Initial TOE and the activated additional code. Identification Data shall be protected in integrity.

OSP.ADDITIONAL_CODE_SIGNING

The additional code has to be signed with a cryptographic key according to relevant standard and the generated signature is associated to the additional code.
The additional code signature must be checked during loading to assure its authenticity and integrity and to assure that loading is authorized on the TOE.
The cryptographic key used to sign the additional code shall be of sufficient quality and its generation shall be appropriately secured to ensure the authenticity, integrity and confidentiality of the key.

OSP.ADDITIONAL_CODE_ENCRYPTION

The additional code has to be encrypted according to relevant standard in order to ensure its confidentiality when it is transmitted to the TOE for loading and installation.
The encryption key shall be of sufficient quality and its generation shall be appropriately secured to ensure the confidentiality, authenticity and integrity of the key.

5.6.3 OSP associated to PACE Module

Note: OSP naming rules for this module (P.X) is coming from [PP_PACE] and remains unchanged for compatibility reason.

P.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 Applicative users as defined in [ICAO-9303].

2.) They shall implement the terminal parts of the PACE protocol [ICAO-TR-SAC], of the Passive Authentication [ICAO-9303] 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.) The related terminals and their environment shall ensure confidentiality and integrity of respective data handled by them (e.g. confidentiality of PACE passwords, etc.), where it is necessary for a secure operation of the TOE.

Application note: Applicative user is travel document holder in MTRD context.

P.Personalisation Personalisation of the applicative data by authorized issuing actor only

The issuer* guarantees the correctness of the user data to be included in TOE in Personalisation phase. In particular, the issuer* guarantees user data are consistent with respect of the end user of the TOE.

Application note: For MRTD application, the issuer is here “issuing State or Organisation”, the user data includes at least, “the biographical data, the printed portrait and the digitized portrait, the biometric reference data and other data of the logical travel document” and the end user is “the travel document holder”. The personalisation of the travel document for the holder is performed by an agent authorized by the issuing State or Organisation only.

P.Manufact Manufacturing of the TOE with Initialization Data for application.

The Initialization Data are written by the IC Manufacturer to identify the IC uniquely. The FF Manufacturer writes the Pre-personalisation Data which contains at least the Personalisation Agent Key.

P.Pre-Operational Pre-operational handling of the TOE and associated applications

1.) The Issuer issues the TOE and associated applications (e.g. travel document) and approves it using the terminals complying with all applicable laws and regulations.

2.) The Issuer guarantees correctness of the user data (amongst other of those, concerning the application user (e.g.travel document holder) and of the TSF-data permanently stored in the TOE[1].

3.) The Issuer uses only such TOE's technical components (IC) which enable traceability of the TOE and associated applications (e.g. travel documents) in their manufacturing and issuing life cycle phases, i.e. before they are in the operational phase.

If the Issuer authorises a Personalisation Agent to personalise the TOE and associated applications (e.g. travel documents) for application user (e.g. travel document holder), the Issuer has to ensure that the Personalisation Agent acts in accordance with the Issuer's policy.

5.7 ASSUMPTIONS

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

5.7.1 Assumptions from Java Card System Protection Profile – Open Configuration

A.CAP_FILE

CAP Files loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM3], §3.3) outside the API.

A.DELETION

Deletion of applets through the card manager is secure. Refer to #.DELETION for details on this assumption.

A.VERIFICATION

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

5.7.2 Assumptions associated to PACE Module

A.Insp_Sys Inspection Systems for global interoperability

The Extended Inspection System (EIS) for global interoperability (i) implements at least the terminal part of PACE [ICAO-TR-SAC]. If several protocols are supported by the EIS, PACE secure channel must be established and applicative data (e.g. the logical travel document) must be transferred under PACE. Other operations may be done when additional protocols are supported by the terminal.

[1] cf. Table 4 and Table 5 above

5.7.3 Assumptions related to path loading

A.OS-UPDATE-EVIDENCE

For additional code loaded pre-issuance, it is assumed that:

- Evaluated technical and/or audited organizational measures have been implemented to ensure that the additional code:
  (1) has been issued by the genuine OS Developer
  (2) has not been altered since it was issued by the genuine OS Developer.

For additional code loaded post-issuance, it is assumed that the OS Developer provides digital evidence to the TOE in order to prove the following:
(1) he is the genuine developer of the additional code and
(2) the additional code has not been modified since it was issued by the genuine OS Developer.

A.SECURE_ACODE_MANAGEMENT

It is assumed that:

- The Key management process related to the OS Update capability takes place in a secure and audited environment.
- The cryptographic keys used by the cryptographic operations are of strong quality and appropriately secured to ensure confidentiality, authenticity and integrity of those keys.

6 SECURITY OBJECTIVES

6.1 SECURITY OBJECTIVES FOR THE TOE

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

6.1.1 Security objectives for the TOE from Java Card System Protection Profile – Open Configuration

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

6.1.1.1 Identification

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

6.1.1.2 Execution

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

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

O.GLOBAL_ARRAYS_INTEG
The TOE shall ensure that 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.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.

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

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

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

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

6.1.1.3 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. 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.

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 the 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.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.

6.1.1.4 Object deletion

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

6.1.1.5 Applet management

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

O.LOAD
The TOE shall ensure that the loading of a CAP file into the card is safe.
Besides, for codes loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application CAP file by the verification authority. This verification by the TOE shall occur during the load or late 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 CAP files sent to the card. He may also try to send one of its own applications as if it came from the card issuer. Thus, this objective is intended to ensure the integrity and authenticity of loaded CAP files.

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

6.1.1.6 SCP

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

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

O.SCP.SUPPORT
The SCP shall support the TSFs of the TOE.
This security objective of the TOE refers to the security aspect 2, 3, 4 and 5 of #.SCP
(2) It does not allow the TSFs to be bypassed or altered and does not allow access to other low-level functions than those made available by the CAP file of the API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System.
(3) It provides secure low-level cryptographic processing to the Java Card System.
(4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism.
(5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low-level control accesses (segmentation fault detection).

O.SCP.IC
The SCP shall provide all IC security features against physical attacks.
This security objective for of the TOE refers to the point (7) of the security aspect #.SCP:
It is required that the IC is designed in accordance with a well-defined set of policies and Standards (likely specified in another protection profile), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

6.1.1.7 CMGR

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

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

6.1.2 Security objectives for the TOE from PACE Module

This section describes the security objectives for the TOE addressing the aspects of identified threats to be countered by the PACE Module of TOE and organisational security policies to be met by the PACE Module of TOE.
Note: TOE objectives naming rules for this module (OT.X) is coming from [PP_PACE] and remains unchanged for compatibility reason.

OT.AC_Pers Access Control for Personalisation of TOE and Applicative data
The TOE must ensure that the TOE and Application data requiring PACE usage* 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 travel document data in EF.DG1 to EF.DG16) and associated TSF data may be written only during and cannot be changed after personalisation phase.

Application note: Application data requiring PACE usage* for MRTD is PACE data, and MTRD data as logical travel document data in EF.DG1 to EF.DG16, the Document Security Object according to LDS [ICAO-9303]).

OT.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.

OT.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[i].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).

OT.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.

OT.Identification Identification 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 application data requiring PACE usage (e.g. travel document for MRTD). The storage of the Pre-Personalisation data includes writing of the Personalisation Agent Key(s).

OT.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.

OT.Prot_Inf_Leak Protection against Information Leakage
The TOE must provide protection against disclosure of confidential User Data or/and TSF-data stored and/or processed by the TOE

- 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.

Application note: This objective pertains to measurements with subsequent complex signal processing due to normal operation of the TOE or operations enforced by an attacker.

OT.Prot_Phys_Tamper Protection against Physical Tampering
The TOE must provide protection of confidentiality and integrity of the User Data, the TSF-data and the TOE's Embedded Software by means of

- measuring through galvanic contacts representing a direct physical probing on the chip's 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 electrical charges (using tools used in solid-state physics research and IC failure analysis),
- manipulation of the hardware and its security functionality, 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 functionality.

OT.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 (esp. electromagnetic) fields, voltage (on any contacts), clock frequency or temperature.

The following TOE security objectives address the aspects of identified threats to be countered involving TOE's environment.

Other security objectives for TOE from [PP_EAC2] are specific to travel document and are not copied here.

6.1.3 Additional objectives

6.1.3.1 Objectives of additional services provided to applications by the TOE

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

O.SpecificAPI

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

6.1.3.2 *Objectives to patch loading and activation*

Security Target of a TOE embedding a Loader shall include the following Security Objectives.

O.SECURE_LOAD_ACODE

The TOE shall check an evidence of authenticity and integrity of the additional code to be loaded.

The TOE enforces that only an allowed version of the additional code can be loaded. The TOE shall forbid the loading of an additional code not intended to be assembled with the TOE.

During the loading of the additional code, the TOE shall remain secure.

O.SECURE_AC_ACTIVATION

Activation of the additional code and update of the Identification Data shall be performed at the same time in an atomic way. All the operations needed for the code to be able to operate as in the Updated TOE shall be completed before activation.

If the atomic activation is successful, then the resulting product is the Updated TOE, otherwise (in case of interruption or incident which prevents the forming of the Updated TOE), the TOE shall preserve a secure state.

O.TOE_IDENTIFICATION

The TOE provides means to store Identification Data in its non-volatile memory and guarantees the integrity of these data.

After atomic activation of the additional code, the Identification Data of the Updated TOE allows identifications of both the Initial TOE and additional code.

The user must be able to uniquely identify Initial TOE and additional code(s) which are embedded in the Updated TOE.

O.CONFID-OS-UPDATE.LOAD

The TOE shall decrypt the additional code prior installation.

Application Note: Confidentiality protection must be enforced when the additional code is transmitted to the TOE for loading (See OE.OS-UPDATE-ENCRYPTION). Confidentiality protection can be achieved either through direct encryption of the additional code, or by means of a trusted path ensuring the confidentiality of the communication to the TOE.

6.2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

6.2.1 Security Objectives for the Operational Environment from Java Card System Protection Profile – Open Configuration

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

OE.VERIFICATION

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

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

OE.CAP_FILE

No CAP file loaded post-issuance shall contain native methods.

OE.CODE-EVIDENCE

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

6.2.2 Security Objectives for the Operational Environment from PACE Module

OE.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 travel document). The inspection system will prevent eavesdropping to their communication with the TOE before secure messaging is successfully established.

OE.Personalisation Personalisation of TOE and application data requiring PACE usage

The Issuer must ensure that the Personalisation Agents acting on his behalf (i) establish the correct identity of the applicative user (e.g. travel 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 travel document), (ii) biometric reference data of the travel document holder, the initial TSF data, (the Document Security Object defined in [ICAO-9303] (in the role of a DS).

OE.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 travel document holders as defined in as defined in [ICAO-9303].

2.) The related terminals implement the terminal parts of the PACE protocol [ICAO-TR-SAC], of the Passive Authentication [ICAO-TR-SAC] (by verification of the signature of the Document Security Object) and use them in this order (This order is commensurate with [ICAO-TR-SAC]. 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 securely store the Country Signing Public Key and the Document Signer Public Key (in form of C_{CSCA} and C_{DS}) in order to enable and to perform Passive Authentication of the travel document (determination of the authenticity of data groups stored in the travel document, [ICAO-9303]).

5.) 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.User_Obligations User Obligations

The application user (e.g. travel 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.

Other security objectives for Operational environment from [PP_EAC2] are specific to travel document and are not copied here.

6.2.3 Supplementary security objectives for the operational environment

The following security objectives for the operational environment shall also be considered for the present evaluation:

OE.OS-UPDATE-EVIDENCE	For additional code loaded pre-issuance, evaluated technical measures implemented by the TOE or audited organizational measures must ensure that the additional code (1) has been issued by the genuine OS Developer (2) has not been altered since it was issued by the genuine OS Developer. For additional code loaded post-issuance, the OS Developer shall provide digital evidence to the TOE that (1) he is the genuine developer of the additional code and (2) the additional code has not been modified since it was issued by the genuine OS Developer.
OE.OS-UPDATE-ENCRYPTION	For additional code loaded post-issuance, the OS Developer shall encrypt the additional code so that its confidentiality is ensured when it is transmitted to the TOE for loading and installation.
OE.SECURE_ACODE_MANAGEMENT	Key management processes related to the OS Update capability shall take place in a secure and audited environment. The key generation processes shall guarantee that cryptographic keys are of sufficient quality and appropriately secured to ensure confidentiality, authenticity and integrity of the keys.

6.3 SECURITY OBJECTIVES RATIONALE

6.3.1 Security objectives rationale from JCS Protection Profile – Open Configuration

	O.SID	O.OPERATE	O.RESOURCES	O.FIREWALL	O.NATIVE	O.REALLOCATION	O.GLOBAL_ARRAYS_CONFID	O.GLOBAL_ARRAYS_INTEG	O.ALARM	O.TRANSACTION	O.CIPHER	O.RNG	O.PIN_MNGT	O.KEY_MNGT	O.OBJ_DELETION	O.INSTALL	O.LOAD	O.DELETION	O.SCP.RECOVERY	O.SCP.SUPPORT	O.SCP.IC	O.CARD-MANAGEMENT	O.ARRAY_VIEWS_CONFID	O.ARRAY_VIEWS_INTEG	O.SpecificAPI	OE.VERIFICATION	OE.CAP_FILE	OE.CODE-EVIDENCE	OE.OS-UPDATE-EVIDENCE	OE.OS-UPDATE-ENCRYPTION	OE.SECURE_ACODE_MANAGEMENT	O.SECURE_LOAD_ACODE	O.SECURE_AC_ACTIVATION	O.TOE_IDENTIFICATION	O.CONFID-OS-UPDATE.LOAD
T.CONFID-JCS-CODE					X																	X				X									
T.CONFID-APPLI-DATA	X	X		X		X	X		X	X	X	X	X	X					X	X		X	X			X									
T.CONFID-JCS-DATA	X	X		X					X										X	X		X				X									
T.INTEG-APPLI-CODE					X																	X				X		X							
T.INTEG-JCS-CODE					X																	X				X		X							
T.INTEG-APPLI-DATA	X	X		X		X		X	X	X	X	X	X	X					X	X		X		X		X		X							
T.INTEG-JCS-DATA	X	X		X					X										X	X		X				X		X							
T.INTEG-APPLI-CODE.LOAD																	X					X						X							
T.INTEG-APPLI-DATA.LOAD																	X					X						X							
T.SID.1	X			X			X	X								X						X													
T.SID.2	X	X		X												X			X	X															
T.EXE-CODE.1				X																						X									
T.EXE-CODE.2																						X													
T.NATIVE					X																	X				X	X								
T.RESOURCES		X	X													X			X	X															
T.INSTALL																X	X					X													
T.DELETION																		X				X													
T.OBJ-DELETION															X																					
T.PHYSICAL																					X															
T.UNAUTHORIZED_TOE_CODE_UPDATE																															X				
T.FAKE-SGNVER-KEY																															X				
T.WRONG-UPDATE-STATE																																X	X		
T.INTEG-OS-UPDATE_LOAD																															X				
T.CONFID-OS-UPDATE_LOAD																																			X
OSP.VERIFICATION																	X									X									
OSP.SpecificAPI																									X										
OSP.RNG																	X																		
OSP.ATOMIC_ACTIVATION																																X			
OSP.TOE_IDENTIFICATION																																	X		
OSP.ADDITIONAL_CODE_SIGNING																															X				
OSP.ADDITIONAL_CODE_ENCRYPTION																														X					X
A.CAP_FILE																											X								
A.DELETION																						X													
A.VERIFICATION																										X		X							
A.OS-UPDATE-EVIDENCE																													X						
A.SECURE_ACODE_MANAGEMENT																															X				

Table 6: Threats, OSP, Assumptions vs Security Objectives

6.3.1.1 Threats

6.3.1.1.1 Confidentiality

T.CONFID-JCS-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of those instructions enables reading a piece of code, no Java Card applet can therefore be executed to disclose a piece of code. Native applications are also harmless because of the objective (O.NATIVE), so no application can be run to disclose a piece of code.
The (#.VERIFICATION) security aspect is addressed in this ST by the objective for the environment OE.VERIFICATION.
The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

T.CONFID-APPLI-DATA This threat is countered by the security objective for the operational environment regarding bytecode verification (OE.VERIFICATION). It is also covered by the isolation commitments stated in the (O.FIREWALL) objective. It relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.
As applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER, 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.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) shall contribute in covering this threat by controlling the sharing of the global PIN between the applets.
Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The disclosure of such data is prevented by the security objective O.GLOBAL_ARRAYS_CONFID.
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-DATA This threat is covered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) security objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

6.3.1.1.2 Integrity

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

T.INTEG-JCS-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective (O.NATIVE), so no application can be run to disclose or modify a piece of code.
The (#.VERIFICATION) security aspect is addressed in this configuration by the objective for the environment OE.VERIFICATION.
The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity.

T.INTEG-APPLI-DATA This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (O.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives O.CARD-MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. The objectives O.SCP.RECOVERY and O.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.
Concerning the confidentiality and integrity of application sensitive data, as applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (O.CIPHER, 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.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) is also concerned.
Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The integrity of the information stored in that buffer is ensured by the objective O.GLOBAL_ARRAYS_INTEG.
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.

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

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

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

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

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

6.3.1.1.4 Unauthorized execution

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

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

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

6.3.1.1.5 Denial of service

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

6.3.1.1.6 Card management

T.INSTALL This threat is covered by the security objective O.INSTALL which ensures that the installation of an applet performs as expected and the security objectives O.LOAD which ensures that the loading of a CAP file into the card is safe.
The objective O.CARD-MANAGEMENT controls the access to card management functions and thus contributes to cover this threat.

T.DELETION This threat is covered by the O.DELETION security objective which ensures that both applet and CAP file deletion perform as expected.
The objective O.CARD-MANAGEMENT controls the access to card management functions and thus contributes to cover this threat.

6.3.1.1.7 Services

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

6.3.1.1.8 Miscellaneous

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

6.3.1.1.9 Patch loading

T.UNAUTHORIZED_TOE_CODE_UPDATE This threat is covered by the O.SECURE_LOAD_ACODE security objective that ensures the authenticity and the integrity of the additional code. It ensure also that that only the allowed code will be load in a secure process.

T.FAKE-SGNVER-KEY This threat is covered by the O.SECURE_LOAD_ACODE security objective which ensures the authenticity and the integrity of the additional code to avoid loading malicious additional code.

T.WRONG-UPDATE-STATE This threat is covered by the O.SECURE_AC_ACTIVATION and O.TOE_IDENTIFICATION security objective that ensures that the update state stay secure during all the loading process

T.INTEG-OS-UPDATE_LOAD This threat is covered by the O.SECURE_LOAD_ACODE security objective that ensures the authenticity and the integrity of the additional code.

T.CONFID-OS-UPDATE_LOAD This threat is covered by the O.CONFID-OS-UPDATE.LOAD security objective that ensures the confidentiality of the additional code when transmitted until installation.

6.3.1.2 *Organizational Security Policies*

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

This policy is also upheld by the security objective of the environment OE.CODE-EVIDENCE which ensures that evidences exist that the application code has been verified and not changed after verification, and by the security objective for the TOE O.LOAD which shall ensure that the loading of a CAP file into the card is safe.

6.3.1.3 *Additional Organizational Security Policies*

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

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

OSP.ADDITIONAL_CODE_ENCRYPTION is enforced by the TOE security objective of the environment OE.OS-UPDATE-ENCRYPTION which ensure the confidentiality of the additional code

OSP.ADDITIONAL_CODE_SIGNING is enforced by the TOE security objective of the environment O.SECURE_LOAD_ACODE which ensure the integrity of the additional code

OSP.ATOMIC_ACTIVATION is enforced by the TOE security objective of the environment O.SECURE_AC_ACTIVATION which ensure the atomicity of the activation of the additional code

OSP.TOE_IDENTIFICATION is enforced by the TOE security objective of the environment O.TOE_IDENTIFICATION which ensure the identification of the additional code

6.3.1.4 *Assumptions*

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.

A.DELETION The assumption A.DELETION is upheld by the security objective O.CARD-MANAGEMENT which controls the access to card management functions such as deletion of applets.

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

6.3.1.5 *Additional Assumptions related to patch loading*

A.OS-UPDATE-EVIDENCE This assumption is upheld by the security objective on the operational environment OE.OS-UPDATE-EVIDENCE that guarantees that the additional code has been issued by the genuine OS Developer, has not been altered since it was issued by the genuine OS Developer.

A.SECURE_ACODE_MANAGEMENT This assumption is upheld by the security objective on the operational environment OE.SECURE_ACODE_MANAGEMENT that guarantees that cryptographic keys are of sufficient quality and appropriately secured to ensure confidentiality, authenticity and integrity of the keys.

6.3.2 Security objectives rationale for PACE Module

6.3.2.1 Threats

The following table provides an overview for security objectives coverage.

	OT.AC_Pers	*OT.Data_Integrity*	*OT.Data_Authenticity*	*OT.Data_Confidentiality*	*OT.Prot_Abuse-Func*	*OT.Prot_Inf_Leak*	*OT.Identification*	*OT.Prot_Phys-Tamper*	*OT.Prot_Malfuntion*	*OE.Prot_Logical_Data*	*OE.Personalisation*	*OE.Terminal*	*OE.User_Obligations*
T.Skimming[2]		X	X	X									X
T.Eavesdropping				X									
T.Abuse-Func					X								
T.Information_Leakage						X							
T.Phys-Tamper								X					
T.Malfunction									X				
T.Forgery	X	X	X		X			X			X	X	

Table 7: Threats vs Security Objectives for PACE Module

The threat **T.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 **OT.Data_Integrity, OT.Data_Authenticity** and **OT.Data_Confidentiality** through the PACE authentication. The objective **OE.User_Obligations** ensures that a PACE session can only be established either by the application user itself (e.g. travel document holder for MRTD) or by an authorised person or device, and, hence, cannot be captured by an attacker.

The threat **T.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 **OT.Data_Confidentiality** through a trusted channel based on the PACE authentication.

The threat **T.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 **OT.AC_Pers** requires the TOE to limit the write access for the TOE and applicative data to the trustworthy Personalisation Agent (cf. **OE.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 **OT.Data_Integrity** and **OT.Data_Authenticity**, respectively. The objectives **OT.Prot_Phys-Tamper** and **OT.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.Terminal** to contribute to secure exchange between the TOE and the terminal.

The threat **T.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 **OT.Prot_Abuse-Func** ensures that the usage of functions having not to be used in the operational phase is effectively prevented.

The threats **T.Information_Leakage**, **T.Phys-Tamper** and **T.Malfunction** are typical for integrated circuits like smart cards under direct attack with high attack potential. The protection of the TOE against these threats is obviously addressed by the directly related security objectives **OT.Prot_Inf_Leak, OT.Prot_Phys-Tamper** and **OT.Prot_Malfunction**, respectively.

[2] Threats and assumptions included from the claimed PACE-PP [7] are marked *in italic letters.* They are listed for the complete overview of threats and assumptions.

6.3.2.2 Organizational Security Policies and Assumptions

	OT.AC_Pers	OT.Data_Integrity	OT.Data_Authenticity	OT.Data_Confidentiality	OT.Prot_Abuse-Func	OT.Prot_Inf_Leak	OT.Identification	OT.Prot_Phys-Tamper	OT.Prot_Malfuntion	OE.Prot_Logical_Data	OE.Personalisation	OE.Terminal	OE.User_Obligations
P.Personalisation	X						X				X		
P.Manufact							X						
P.Pre-Operational	X						X				X		
P.Terminal												X	
A.Insp_Sys										X			

Table 8: OSP and Assumptions vs Security Objectives for PACE Module

The OSP **P.Personalisation** addresses the (i) the enrolment of the logical travel document by the Personalisation Agent as described in the security objective for the TOE environment **OE.Personalisation**, and (ii) the access control for the user data and TSF data as described by the security objective **OT.AC_Pers.** Note the manufacturer equips the TOE with the Personalisation Agent Key(s) according to **OT.Identification** “Identification and Authentication of the TOE”.

The OSP **P.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 **OT.Identification**.

The OSP **P.Pre-Operational** is enforced by the following security objectives: **OT.Identification** is affine to the OSP’s property ‘traceability before the operational phase’;**OT.AC_Pers** and **OE.Personalisation** together enforce the OSP’s properties ‘correctness of the User- and the TSF-data stored’ and ‘authorisation of Personalisation Agents’.

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

A.Insp_Sys is covered by OE.Prot_Logical_Data requiring the Inspection System to protect the TOE and application data (e.g. the logical travel document data) during the transmission and the internal handling.

6.3.2.3 Compatibility between objectives of the TOE and objectives of [IFX-IC]

6.3.2.3.1 Compatibility between objectives for the TOE

O.SID, O.OPERATE, O.RESOURCES, O.FIREWALL, O.NATIVE, O.REALLOCATION, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.ARRAY_VIEWS_CONFID, O. ARRAY_VIEWS_INTEG, O.ALARM; O.TRANSACTION, O.PIN-MNGT, O.KEY-MNGT, O.OBJ-DELETION, O.INSTALL, O.LOAD, O.DELETION, O.CIPHER are objectives specific to the Java Card platform and they do no conflict with the objectives of [IFX-IC].

O.CARD-MANAGEMENT, O.SpecificAPI, O.SECURE_LOAD_ACODE, O.SECURE_AC_ACTIVATION, O.TOE_IDENTIFICATION and O.CONFID-OS-UPDATE.LOAD are objectives added to this platform it does no conflict with the objectives of [IFX-IC].

O.RNG added to this platform is included in the following objectives of [IFX-IC]: O.RND
O.SCP.IC is included in the following objectives of [IFX-IC]: O.Phys-Manipulation, O.Phys-Probing, O.Malfunction O.Leak-Inherent O.Leak-Forced O.Abuse-Func.
O.SCP.RECOVERY is included in the following objectives of [IFX-IC]: O.Leak-Inherent, O.Leak-Forced, O.Malfunction
O.SCP.SUPPORT is included in the following objectives of [IFX-IC]: O.Mem-Access, O.Prot_TSF_Confidentiality

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

6.3.2.3.2 Compatibility between objectives for the environment

OE.Personalisation, OE.VERIFICATION, OE.CODE-EVIDENCE and OE.CAP_FILE are objectives specific to the Java Card platform and they do no conflict with the objectives of [IFX-IC].

OE.OS-UPDATE-EVIDENCE, OE.OS-UPDATE-ENCRYPTION and OE.SECURE_ACODE_MANAGEMENT are objectives for environment added to this platform it does no conflict with the objectives of [IFX-IC].

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

6.3.2.4 Compatibility between objectives of PACE Module and [IFX-IC]

6.3.2.4.1 Compatibility between objectives for the TOE

OT_AC_Pers is specific to the current document and it does no conflict with the objectives of [IFX-IC].
OT.Data_Confidentiality; OT.Data_Integrity and OT.Data_Authenticity are linked in O.Phys-Manipulation and O.RNG used for cryptographic operations.
OT.Identification is linked to O.Identification.
OT.Prot_Abuse-Func is linked in O.Abuse-Func.
OT.Prot_Inf_Leak is linked in O.Leak-Inherent and O.Leak-Forced
OT.Prot_Phys-Tamper is linked in O.Phys-Manipulation.
OT.Prot_Malfunction is linked in O.Malfunction.

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

6.3.2.4.2 Compatibility between objectives for the environment

[IFX-IC] Objectives	IrOE	CfPOE	SgOE	Description
OE.Resp-Appl		X OE.Personalization		Treatment of User Data of the Composite TOE
OE.Process-Sec-IC		X OE.Personalization		Protection during composite product manufacturing
OE.Lim_Block_Loader	X			Limitation of capability and blocking the Loader
OE.Loader_Usage (only applicable , if Flash Loader active and TOE is ordered with configuration option EA unavailable)	X			Secure communication and usage of the Loader
OE.TOE_Auth (applicable , if Flash Loader active and TOE is ordered with configuration option EA unavailable)	X			External entities authenticating of the TOE

Table 9 Compatibility between environment objectives of PACE Module and [IFX-IC]

OE.Prot_Logical_Data, OE.Terminal, OE.User_Obligations, are specific to this TOE and they do no conflict with the objectives of [ST-IC].

We can therefore conclude that the objectives for the environment of PACE module and [IFX-IC] are consistent.

7 EXTENDED COMPONENTS DEFINITION

7.1 EXTENDED COMPONENTS DEFINITION FROM PP_JCS

7.1.1 Definition of the Family FCS_RNG

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

FCS_RNG Generation of random numbers

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

Component levelling:

FCS_RNG Generation of random numbers	1

FCS_RNG.1	Generation of random numbers requires that random numbers meet a defined quality metric.
Management:	FCS_RNG.1 There are no management activities foreseen.
Audit:	FCS_RNG.1 There are no actions defined to be auditable.

FCS_RNG.1 Random number generation

Hierarchical to:	No other components
Dependencies:	No dependencies
FCS_RNG.1.1	The TSF shall provide a [selection: physical, non-physical true, deterministic, hybrid physical, hybrid deterministic] random number generator [selection: DRG.2, DRG.3, DRG.4, PTG.2, PTG.3, NTG.1] [AIS20] [AIS31] that implements: [assignment: list of security capabilities].
FCS_RNG.1.2	The TSF shall provide random numbers that meet [assignment: a defined quality metric].

Refinement for AIS31 DRG4 compliancy:

The TSF shall provide a hybrid deterministic random number generator that implements:
(DRG.4.1) The internal state of the RNG shall use PTRNG of class PTG.2 as random source.
(DRG.4.2) The RNG provides forward secrecy.
(DRG.4.3) The RNG provides backward secrecy even if the current internal state is known.
(DRG.4.4) The RNG provides enhanced forward secrecy after calling the re-seed function that acts as a refreshing done at each random generation.
(DRG.4.5) The internal state of the RNG is seeded by an internal entropy source, PTRNG of class PTG.2.
The TSF shall provide random numbers that meet:
(DRG.4.6) The RNG generates output for which 2^35 strings of bit length 128 are mutually different with probability equal to (1 – 1/2^58).
(DRG.4.7) Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.

7.2 EXTENDED COMPONENTS DEFINITION FROM PACE MODULE

This security target uses components defined as extensions to CC part 2. Some of these components are defined in protection profile [PP-IC-0084], others are defined in the protection profile [PP-EAC2].

7.2.1 Definition of the Family FMT_LIM

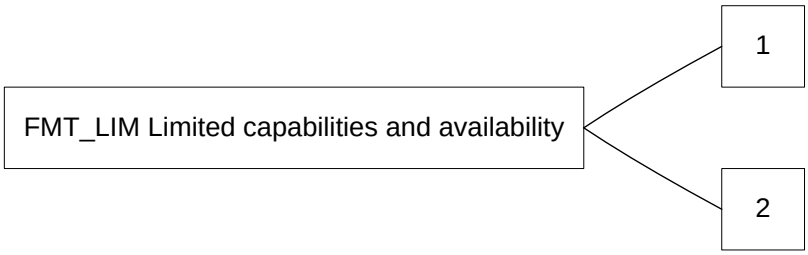
The family FMT_LIM describes the functional requirements for the Test Features of the TOE. The new functional requirements were defined in the class FMT because this class addresses the management of functions of the TSF. The examples of the technical mechanism used in the TOE show that no other class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

The family “Limited capabilities and availability (FMT_LIM)” is specified as follows.

FMT_LIM Limited capabilities and availability

Family behavior
This family defines requirements that limit the capabilities and availability of functions in a combined manner. Note that FDP_ACF restricts the access to functions whereas the Limited capability of this family requires the functions themselves to be designed in a specific manner.

Component leveling:



FMT_LIM.1	Limited capabilities requires that the TSF is built to provide only the capabilities (perform action, gather information) necessary for its genuine purpose.
FMT_LIM.2	Limited availability requires that the TSF restrict the use of functions (refer to Limited capabilities (FMT_LIM.1)). This can be achieved, for instance, by removing or by disabling functions in a specific phase of the TOE’s life-cycle.
Management:	FMT_LIM.1, FMT_LIM.2 There are no management activities foreseen.
Audit:	FMT_LIM.1, FMT_LIM.2 There are no actions defined to be auditable.

The TOE Functional Requirement “Limited capabilities (FMT_LIM.1)” is specified as follows.

FMT_LIM.1 Limited capabilities

Hierarchical to:	No other components
Dependencies:	FMT_LIM.2 Limited availability.

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*].

The TOE Functional Requirement “Limited availability (FMT_LIM.2)” is specified as follows.

FMT_LIM.2 Limited availability

Hierarchical to:	No other components
Dependencies:	FMT_LIM.1 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*].

Application note: The functional requirements FMT_LIM.1 and FMT_LIM.2 assume that there are two types of mechanisms (limited capabilities and limited availability) which together shall provide protection in order to enforce the policy. This also allows that

(i) the TSF is provided without restrictions in the product in its user environment but its capabilities are so limited that the policy is enforced

or conversely

(ii) the TSF is designed with test and support functionality that is removed from, or disabled in, the product prior to the Operational Use Phase.

The combination of both requirements shall enforce the policy.

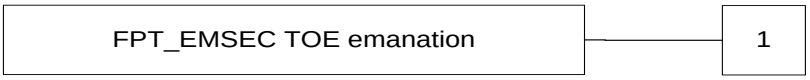
7.2.2 Definition of the Family FPT_EMS

The sensitive 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 TOE 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, etc. This family describes the functional requirements for the limitation of intelligible emanations which are not directly addressed by any other component of CC part 2 [CC-2].

The family “TOE Emanation (FPT_EMS)” is specified as follows.

Family behaviour
This family defines requirements to mitigate intelligible emanations.

Component levelling:



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.

Management:	FPT_EMS.1 There are no management activities foreseen.
Audit:	FPT_EMS.1 There are no actions defined to be auditable.

FPT_EMS.1 TOE Emanation

Hierarchical to:	No other components
Dependencies:	No dependencies.

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*] and [assignment: *list of types of user 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*] and [assignment: *list of types of user data*].

8 SECURITY REQUIREMENTS

8.1 SECURITY FUNCTIONAL REQUIREMENTS

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

8.1.1 Security Functional Requirements from PP Java Card System – Open configuration

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

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

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

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

Subject	Description
S.ADEL	The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([JCRE22], §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.
S.CAD	The CAD represents off-card entity that communicates with the S.INSTALLER.
S.INSTALLER	The installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the loading of CAP files and installation of applets.
S.JCRE	The runtime environment on which Java programs in a smart card are executed.
S.JCVM	The bytecode interpreter that enforces the firewall at runtime.
S.LOCAL	Operand stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.
S.MEMBER	Any object's field, static field or array position.
S.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 a 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.

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

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

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

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

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

Security attribute	Description/Value
Active Applets	The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels.
Applet Selection Status	"Selected" or "Deselected"
Applet's version number	The version number of an applet indicated in the export file
Class	Identifies the implementation class of the remote object.
Context	CAP file AID, or "Java Card RE"
Currently Active Context	CAP file AID, or "Java Card RE"
Dependent package AID	Allows the retrieval of the package AID and Applet's version number ([JCVM3], §4.5.2).
ExportedInfo	Boolean (Indicates whether the remote object is exportable or not).
Identifier	The Identifier of a remote object or method is a number that uniquely identifies a remote object or method, respectively.
LC Selection Status	Multiselectable, Non-multiselectable or "None".
LifeTime	CLEAR_ON_DESELECT or PERSISTENT (*).
Owner	The Owner of an object is either the applet instance that created the object or the CAP file (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the CAP file). The owner of a remote object is the applet instance that created the object.
CAP File AID	The AID of a CAP file.
Package AID	The AID of each package indicated in the export file
Registered applets	The set of AID of the applet instance registered on the card
Resident CAP files	The set of AIDs of the CAP files already loaded on the card.
Selected Applet Context	CAP File AID, or "None"
Sharing	Standards, SIO, Arraw view, Java Card RE entry point, or global array
Static References	Static fields of a CAP file may contain references to objects. The Static References attribute records those references.

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

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

Operation	Description
OP.ARRAY_ACCESS(O.JAVAOBJECT, field)	Read/Write an array component.
OP.ARRAY_LENGTH (O.JAVAOBJECT, field)	Get length of an array component.
OP.ARRAY_AASTORE(O.JAVAOBJECT, field)	Store into reference array component
OP.ARRAY_T_ALOAD(O.JAVAOBJECT, field)	Read from an array component
OP.ARRAY_T_ASTORE(O.JAVAOBJECT, field)	Write to an array component
OP.CREATE(Sharing, LifeTime) (*)	Creation of an object (new or makeTransient or createArrawView call).
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed applet and its objects, either logically or physically.
OP.DELETE_CAP_FILE(O.CODE_CAP_FILE,...)	Delete a CAP file, either logically or physically.
OP.DELETE_CAP_FILE_APPLET(O.CODE_CAP_FILE,...)	Delete a CAP file and its installed applets, either logically or physically.
OP.INSTANCE_FIELD(O.JAVAOBJECT, field)	Read/Write a field of an instance of a class in the Java programming language
OP.INVK_VIRTUAL(O.JAVAOBJECT, method, arg1,...)	Invoke a virtual method (either on a class instance or an array object)
OP.INVK_INTERFACE(O.JAVAOBJECT, method, arg1,...)	Invoke an interface method.
OP.JAVA(...)	Any access in the sense of [JCRE3], §6.2.8. It stands for one of the operations OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW, OP.TYPE_ACCESS.
OP.PUT(S1,S2,I)	Transfer a piece of information I from S1 to S2.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (throw, see [JCRE3],§6.2.8.7)
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	Invoke checkcast or instanceof on an object in order to access to classes (standard or shareable interfaces objects).

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

8.1.1.1 CoreG_LC Security Functional Requirements

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

8.1.1.1.1 Firewall Policy

FDP_ACC.2/FIREWALL Complete access control

FDP_ACC.2.1/FIREWALL The TSF shall enforce the **FIREWALL access control SFP** on **S.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_T_ALOAD
- OP.ARRAY_T_ASTORE
- 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:
Accessing array's components of a static array, and more generally fields and methods of static objects, is an access to the corresponding O.JAVAOBJECT.

FDP_ACF.1/FIREWALL Security attribute based access control

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

Subject/Object	Attributes
S.CAP_FILE	**LC Applet Selection Status**
S.JCVM	**ActiveApplets, Currently Active Context**
S.JCRE	**Selected Applet Context**
O.JAVAOBJECT	**Sharing, Context, LifeTime**

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

- **R.JAVA.1 ([JCRE3]§6.2.8) An S.CAP_FILE may freely perform any of OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW or OP.TYPE_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".**
- **R.JAVA.2 ([JCRE3]§6.2.8) An S.CAP_FILE may freely perform any of OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE or OP.THROW upon any O.JAVAOBJECT whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if O.JAVAOBJECT's Context attribute has the same value as the active context.**
- **R.JAVA.3 ([JCRE3]§6.2.8.10) An 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 ([JCRE3], §6.2.8.6,) An 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 "CAP file AID", only if the invoked interface method extends the Shareable interface and one of the following applies:**

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

  (b) The value of the attribute Selection Status of the CAP file whose AID is "Package AID' is "Non-multiselectable», and either "CAP file AID" is the value of the currently selected applet or otherwise "CAP file AID" does not occur in the attribute ActiveApplets.

- **R.JAVA.5 An S.CAP_FILE may perform an OP.CREATE only if the value of the Sharing parameter(*) is "Standard".**

- **R.JAVA.6 ([JCRE3], §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".**

  Application Note (R.JAVA.4): The initial setting of security attributes ActiveApplets and Selected Applet Context are initialized by SELECT APDU and MANAGE_CHANNEL, which are out of SPM scope. The ActiveApplets and Selected Applet Context are never changed in the VM scope.

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

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

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

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

   Application note: This rule is out of scope of the SPM modelisation because CLEAR_ON_DESELECT objects can be created exclusively in the API, which is also out of scope (Hypothesis 4 of the SPM document [SPM])..

3) **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”.**
4) **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”**
5) **R.JAVA.7 ([JCRE3], §6.2.8.2): S.CAP_FILE performing OP.ARRAY_T_ASTORE of the reference of an O.JAVAOBJECT, or a primitive value when the O.JAVAOBJECT is an array view without ATTR_WRITABLE_VIEW access attribute.**
6) **R.JAVA.8 ([JCRE3], §6.2.8.2):S.CAP_FILE performing OP.ARRAY_T_ALOAD of the reference of an O.JAVAOBJECT, or a primitive value when the O.JAVAOBJECT is an array view without ATTR_READABLE_VIEW access attribute.**

Application note: FDP_ACF.1.4/FIREWALL:

The initial setting of security attribute Selected Applet Context is initialized by SELECT APDU, which is out of SPM scope. Selected Applet Context is never changed in the VM scope.

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.

The deletion of applets is out of scope of this SPM scope.

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 five 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 ([JCRE3], §6.1.3). An object is owned by an applet instance, by the JCRE or by the CAP file library where it has been defined (these latter objects can only be arrays that initialize static fields of CAP file).

([JCRE3], 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 (CAP file AID) of the selected applet. In this policy, the "Selected Applet Context" is the AID of the selected CAP file.
([JCRE3], §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 CAP file" is not the one to which the static method belongs to in this case.
The Java Card platform, version 2.2.x introduces the possibility for an applet instance to be selected on multiple logical channels at the same time, or accepting other applets belonging to the same CAP file being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same CAP file are either all multiselectable or not ([JCVM3], §2.2.5). Therefore, the selection mode can be regarded as an attribute of CAP file. No selection mode is defined for a library CAP file.

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. ([JCRE3], §4).

FDP_IFC.1/JCVM Subset information flow control

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

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

FDP_IFF.1/JCVM Simple security attributes

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

Subject / Information	Description
S.JCVM	Currently active context.

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

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

FDP_IFF.1.3/JCVM The TSF shall enforce **no additional information flow control SFP rules**.

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

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

Application Note:

The storage of temporary Java Card RE-owned objects references is runtime-enforced ([JCRE3], §6.2.8.1-3).

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

FDP_RIP.1/OBJECTS Subset residual information protection

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

FMT_MSA.1/JCRE Management of security attributes

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

Application note:
The modification of the Selected Applet Context is performed in accordance with the rules given in [JCRE3], §4 and [JCVM3], §3.4.

The initial setting of security attribute the Selected Applet Context is initialized by SELECT APDU and MANAGE_CHANNEL, which are out of SPM scope. The the Selected Applet Context is never changed in the VM scope.

FMT_MSA.1/JCVM Management of security attributes

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

Application note:

The modification of the Selected Applet Context is performed in accordance with the rules given in [JCRE3], §4 and [JCVM3], §3.4.

The initial setting of security attribute ActiveApplets is initilized by SELECT APDU and MANAGE_CHANNEL, which are out of SPM scope. The ActiveApplets is never changed in the VM scope.

FMT_MSA.2/FIREWALL_JCVM Secure security attributes

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

FMT_MSA.3/FIREWALL Static attribute initialization

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

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

Application Note:

FMT_MSA.3.1/FIREWALL

- Objects' security attributes of the access control policy are created and initialized at the creation of the object or the subject. Afterwards, these attributes are no longer mutable (FMT_MSA.1/JCRE). At the creation of an object (OP.CREATE), the newly created object, assuming that the FIREWALL access control SFP permits the operation, gets its Lifetime and Sharing attributes from the parameters of the operation; on the contrary, its Context attribute has a default value, which is its creator's Context attribute and AID respectively ([JCRE3], §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 initialization

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

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

FMT_SMR.1/JCRE Security roles

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

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

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

FMT_SMF.1/CORE_LC Specification of Management Functions

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

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

Note: the Selected Applet context is out of scope of the VM functionalities. It is a process that occurs prior to VM start

The initial setting of security attributes ActiveApplets and Selected Applet Context are initilized by SELECT APDU and MANAGE_CHANNEL, which are out of SPM scope. The ActiveApplets and Selected Applet Context are never changed in the VM scope.

8.1.1.1.2 Application Programming Interface

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

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 **[assignment: cryptographic key generation algorithm]** and specified cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignment: list of standards]**.

Iteration	Algorithm	Key size	Standards
/RSA Std	RSA standard key generation	1024, 1536, 2048	ANSI X9.31
/RSA CRT	RSA CRT key generation	1024, 1536, 2048, 4096	ANSI X9.31
/GP	GP session keys	112 (for SCP01, SCP02) 128,192, 256 (for SCP03)	[GP23] (for SCP01, SCP02) [GP23] (for SCP03)
/ECFP	ECC key generation	160, 192, 224, 256, 320, 384, 512, 521	ANSI X9.62
/ECDH	EC Diffie-Hellman	160, 192, 224, 256, 320, 384, 512, 521	ANSI X9.63
/DHGen	DH key generation	1024, 1280 ,1536, 2048, 3072	ANSI X9.42
/DH	DH key exchange	1024, 1280,1536, 2048; 3072	ANSI X9.42
/ML-DSA.KeyGen	Module Lattice Based Digital Signature key generation	1952	NIST FIPS 204

Application note:

- The keys are generated and diversified in accordance with [JCAPI3] specification in classes KeyBuilder and KeyPair (at least Session key generation) and RandomData.

FCS_CKM.2 Cryptographic key distribution

FCS_CKM.2.1 The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **[assignment: cryptographic key distribution method]** that meets the following: **[assignment: list of standards]**.

For Applications:

iteration	Distribution method	standards
/RSA	JC API setkey()	[JCAPI3]
/TDES	JC API setkey()	[JCAPI3]
/AES	JC API setkey()	[JCAPI3]
/ECFP	JC API setkey()	[JCAPI3]
/DH	Thales API setkey()	Thales DIS specification
/ML-DSA	Thales API setkey()	Thales DIS specification

Note: The “/DH, /ML-DSA Thales API setkey()” are part of proprietary API that are not publicly documented (Thales DIS Specification).

For GP:

STORE DATA command, standard [JCAPI3]

PUT KEY command, standard [JCAPI3]

FCS_CKM.3 Cryptographic key access

FCS_CKM.3.1 The TSF shall perform **[assignment: type of cryptographic key access]** in accordance with a specified cryptographic key access method **[assignment: cryptographic key access method]** that meets the following: **[assignment: list of standards]**.

iteration	Key access method	standards
/RSA	JC API getkey()	[JCAPI3]
/TDES	JC API getkey()	[JCAPI3]
/AES	JC API getkey()	[JCAPI3]
/ECFP	JC API getkey()	[JCAPI3]
/DH	Thales API getkey()	Thales DIS specification
/ML-DSA	Thales API getkey()	Thales DIS specification

Note: The "/DH, /ML-DSA Thales API getkey()" are part of proprietary API that are not publicly documented (Thales DIS Specification).

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 **physical irreversible destruction of the stored key value** that meets the following: **No standard**.
Application note:

- The keys are reset in accordance with [JCAPI3] in class Key with the method clearKey(). Any access to a cleared key attempting to use it for ciphering or signing shall throw an exception.

FCS_COP.1 Cryptographic operation

FCS_COP.1.1 The TSF shall perform **[assignment: list of cryptographic operations]** in accordance with a specified cryptographic algorithm **[assignment: cryptographic algorithm]** and cryptographic key sizes **[assignment: cryptographic key sizes]** that meet the following: **[assignment: list of standards]**.

Iteration	operation	algorithm	Key size	Standards
/RSA-SIGN	signature & verification	RSA (STD) RSA CRT	1024, 1152, 1280, 1536 and 2048 3072, 4096	[ISO9796-2] RSA SHA PKCS#1 RSA SHA PKCS#1 PSS
/RSA-CIPHER	Encryption & decryption	RSA (STD) RSA (CRT)	1024, 1152, 1280, 1536, 2048 3072, 4096	[ISO9796-2] RSA SHA PKCS#1 OAEP
/ECC-SIGN	signature & verification	ECC	160, 192, 224, 256, 320, 384, 512, 521	[TR-03111] ECDSA SHA
/TDES-CIPHER	Encryption & decryption	TDES	112 168	[SP800-67] [ISO9797-1] DES NOPAD DES PKCS#5 DES 9797 M1 M2
/AES-CIPHER	Encryption & decryption	AES	128, 192, 256	[FIPS197] AES 128 NOPAD
/AES-CIPHER FAST	Encryption & decryption	AES	128, 192, 256	[FIPS197] AES 128 NOPAD

/TDES-CIPHER FAST	Encryption & decryption	TDES	112 168	[SP800-67] [ISO9797-1] DES NOPAD DES PKCS#5 DES 9797 M1 M2
/TDES-MAC	Signature, Verification	TDES	112 168	[SP800-67] [ISO9797-1] DES MAC ISO9797-1 M1 M2 Alog3 DES MAC NOPAD DES MAC PKCS#5
/TDES-MAC FAST	Signature, Verification	TDES	112 168	[SP800-67] [ISO9797-1] DES MAC ISO9797-1 M1 M2 Alog3 DES MAC NOPAD DES MAC PKCS#5
/AES-MAC	Signature, Verification	AES	128, 192, 256	[FIPS197] AES 128 NOPAD; SP800-38B
/AES-MAC FAST	Signature, Verification	AES	128, 192, 256	[FIPS197] AES 128 NOPAD; SP800-38B
/AES-CMAC	Signature, Verification	AES	128, 192, 256	SP800-38B
/AES-CMAC FAST	Signature, Verification	AES	128, 192, 256	SP800-38B
/SHA	Hashing	Hashing	SHA-1, SHA2-224, SHA2-256, SHA2-384, SHA2-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512 SHAKE256	[FIPS180-4] [FIPS202]

/DH-PACE	Integrited Mapping Generic Mapping	DH	1024, 2048	ISO/IEC JTC1 SC17 WG3/TF5 ‘Supplemental Access Control for Machine Readable Travel Documents’
/ECC-PACE	Integrited Mapping Generic Mapping	ECC	160, 192, 224, 256, 320, 384, 512, 521	ISO/IEC JTC1 SC17 WG3/TF5 ‘Supplemental Access Control for Machine Readable Travel Documents’
HMAC			SHA-1,SHA-224, SHA-256, SHA-384, SHA-512	
OBKG	Key Generation	ECC RSA	160 – 521 1024 – 2048 STD 1024 – 4096 CRT	
ML-DSA	Signature, verification	ML-DSA-65 (Level3)	public key 1952 bytes / private key 4032	FIPS204

FDP_RIP.1/ABORT Subset residual information protection

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

FDP_RIP.1/APDU Subset residual information protection

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

FDP_RIP.1/GlobalArray Subset residual information protection

FDP_RIP.1.1/GlobalArray [Refined]
The TSF shall ensure that any previous information content of a resource is made unavailable upon **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)**.

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 [JCRE3], §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 CAP file must share the transient memory segment if they are concurrently active ([JCRE3], §4.238.

FDP_ROL.1/FIREWALL Basic rollback

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

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 [JCRE3], §7.7, within the bounds of the Commit Capacity ([JCRE3], §7.8), and those described in [JCAPI3]**.

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 [JCAPI3] (see for instance, PIN-blocking, PIN-checking, update of Transient objects).

8.1.1.1.3 Card Security Management

FAU_ARP.1 Security alarms

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

- **throw an exception,**
- **or lock the card session**
- **or reinitialize the Java Card System and its data**

upon detection of a potential security violation.

Refinement:
The TOE detects the following potential security violation:

- CAP file inconsistency
- Applet life cycle inconsistency
- Card Manager life cycle inconsistency
- Card tearing (unexpected removal of the Card out of the CAD) and power failure
- Abortion of a transaction in an unexpected context (see abortTransaction(), [JCAPI3] and ([JCRE3], §7.6.2)
- Violation of the Firewall or JCVM SFPs
- Unavailability of resources
- Array overflow
- Random trap detection

Application Note:

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

FDP_SDI.2/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: **integrity-sensitive data**.

FDP_SDI.2.2/DATA Upon detection of a data integrity error, the TSF shall
- Prevent the use of modified data
- Raise an exception

Application note:
- Although no such requirement is mandatory in the Java Card specification, at least an exception shall be raised upon integrity errors detection on cryptographic keys, PIN values and their associated security attributes. Even if all the objects cannot be monitored, cryptographic keys and PIN objects shall be considered with particular attention by ST authors as they play a key role in the overall security.
- It is also recommended to monitor integrity errors in the code of the native applications and Java Card applets.

For integrity sensitive application, their data shall be monitored (D.APP_I_DATA): applications may need to protect information against unexpected modifications, and explicitly control whether a piece of information has been changed between two accesses. For example, maintaining the integrity of an electronic purse's balance is extremely important because this value represents real money. Its modification must be controlled, for illegal ones would denote an important failure of the payment system.
- A dedicated library could be implemented and made available to developers to achieve better security for specific objects, following the same pattern that already exists in cryptographic APIs, for instance.

FPR_UNO.1 Unobservability

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

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/JCS Failure with preservation of secure state

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

Application note:
- The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE3], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE3]). Behavior of the TOE on power loss and reset is described in [JCRE3], §3.6, and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE3], §3.6.2.

FPT_TDC.1 Inter-TSF basic TSF data consistency

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

FPT_TDC.1.2 The TSF shall use
- **The rules defined in [JCVM3] specification**
- **The API tokens defined in the export files of reference implementation**
- **The rules defined in ISO 7816-6**
- **The rules defined in [GP23] specification**

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.

8.1.1.1.4 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:
- **CAP file AID**
- **Applet's version number**
- **registered applet's AID**
- **applet selection status**

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

FIA_UID.2/AID User identification before any action

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

Application notes:
- By users here it must be understood the ones associated to the CAP file (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 CAP file that is the subject's owner. Means of identification are provided during the loading procedure of the CAP file and the registration of applet instances.
- The role Java Card RE defined in FMT_SMR.1/JCRE is attached to an IT security function rather than to a "user" of the CC terminology. The Java Card RE does not "identify" itself with respect to the TOE, but it is a part of it.

FIA_USB.1/AID User-subject binding

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

FIA_USB.1.2/AID The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users:
- **Initial applet selection is performed as described in [JCRE3]§4**
- **The default applet depends on personalization.**

FIA_USB.1.3/AID The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users:
- **Applet selection is performed after a successful SELECT FILE command as described in [JCRE3]§4**.

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 "CAP file 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).
- **The DELETE and INSTALL APDU commands are out of scope of this SPM. The list of registred applets' AIDs is proven to be not modified during the execution inside the VM.**

FMT_MTD.3/JCRE Secure TSF data

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

8.1.1.2 INSTG Security Functional Requirements

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

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

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

- 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 CAP file, maximal operand stack size and number of local variables for each method, and export and import components (accessibility).

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

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

Application note:

- The format of the CAP file is precisely defined in Sun's specification ([JCVM3]); it contains the user data (like applet's code and data) and the security attribute altogether. Therefore, there is no association to be carried out elsewhere.

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

Application note:

- Each CAP File contains a Version attribute, which is a pair of major and minor version numbers ([JCVM3], §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 ([JCVM3], §4.5.2): the dependent packages Versions and AIDs attributes allow the retrieval of these identifications... Implementation-dependent checks may occur on a case-by-case basis to indicate that package files are binary compatibles. However, package files do have "package Version Numbers" ([JCVM3]) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.

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

CAP file loading is allowed only if, for each dependent CAP file, its AID attribute is equal to a resident CAP file AID attribute, the major (minor) Version attribute associated to the dependent CAP file is lesser than or is equal to the major (minor) Version attribute associated to the resident CAP file ([JCVM3],§4.5.2).

Application note:

- 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 ([JCVM3], §4.4).
- The installation (the invocation of an applet's install method by the installer) is implementation dependent ([JCRE3], §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, [JCRE3], §11).

FMT_SMR.1/Installer Security roles

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

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

FPT_FLS.1/Installer Failure with preservation of secure state

FPT_FLS.1.1/Installer The TSF shall preserve a secure state when the following types of failures occur: **the installer fails to load/install a CAP file/applet as described in [JCRE3] §11.1.5**.

Application Note:

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

FPT_RCV.3/Installer Automated recovery without undue loss

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

Application note:

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

FPT_RCV.3.2/Installer for **[Failure during applet loading, installation and deletion; sensitive data loading]**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

Application note:

- Should the installer fail during loading/installation of a CAP file/applet, it has to revert to a "consistent and secure state". The Java Card RE has some clean up duties as well; see [JCRE3], §11.1.5 for possible scenarios. Precise behavior is left to implementers. This component shall include among the listed failures the deletion of a CAP file/applet. See ([JCRE3], 11.3.4) for possible scenarios. Precise behavior is left to implementers.
- Other events such as the unexpected tearing of the card, power loss, and so on, are partially handled by the underlying hardware platform (see [PP-IC-0084]) 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 functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **[none]** for loss of TSF data or objects under the control of the TSF.

Application note:

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

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

8.1.1.3 ADELG Security Functional Requirements

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

FDP_ACC.2/ADEL Complete access control

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

Refinement:
The operations involved in the policy are:

- OP.DELETE_APPLET,
- OP.DELETE_CAP_FILE,
- OP.DELETE_CAP_FILE_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 CAP files**
O.CODE_CAP_FILE	**CAP file AID, Dependent CAP file AID, Static References**
O.APPLET	**Applet Selection Status**
O.JAVAOBJECT	**Owner, Remote**

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

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

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

R.JAVA.14 ([JCRE3], §11.3.4.2, Applet Instance Deletion). The S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if,

(1) S.ADEL is currently selected,
(2) There is no instance in the context of O.APPLET that is active in any logical channel and
(3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a CAP File P, or ([JCRE3], §8.5) O.JAVAOBJECT is remote reachable.

R.JAVA.15 ([JCRE3], §11.3.4.2, Multiple Applet Instance Deletion). S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if,

(1) S.ADEL is currently selected,
(2) There is no instance in the context of O.APPLET that is active in any logical channel and
(3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is reachable from a CAP file P, or ([JCRE3], §8.5) O.JAVAOBJECT is remote reachable.

R.JAVA.16 ([JCRE3], §11.3.4.3, Applet/Library CAP file Deletion). The S.ADEL may perform OP.DELETE_CAP_FILE upon an O.CODE_CAP_FILE only if,

(1) S.ADEL is currently selected,
(2) no reachable O.JAVAOBJECT, from a CAP file distinct from O.CODE_CAP_FILE that is an instance of a class that belongs to O.CODE_CAP_FILE exists on the card and
(3) there is no resident CAP file on the card that depends on O.CODE_CAP_FILE.

R.JAVA.17 ([JCRE3], §11.3.4.4, Applet CAP file and Contained Instances Deletion). S.ADEL may perform OP.DELETE_CAP_FILE_APPLET upon an O.CODE_CAP_FILE only if,

(1) S.ADEL is currently selected,
(2) no reachable O.JAVAOBJECT, from a CAP file distinct from O.CODE_CAP_FILE, which is an instance of a class that belongs to O.CODE_CAP_FILE exists on the card,
(3) there is no CAP file loaded on the card that depends on O.CODE_CAP_FILE 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 CAP file not being deleted, or ([JCRE3],§8.5) O.JAVAOBJECT is remote reachable.

Application notes:

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

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

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

any subject but the S.ADEL to O.CODE_CAP_FILE or O.APPLET for the purpose of deleting it from the card.

FDP_RIP.1/ADEL Subset residual information protection

FDP_RIP.1.1/ADEL The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **applet instances and/or CAP file 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/CAP file deletion are described in [JCRE3], §11.3.4.1, §11.3.4.2 and §11.3.4.3.

FMT_MSA.1/ADEL Management of security attributes

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

FMT_MSA.3/ADEL Static attribute initialization

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

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

FMT_SMF.1/ADEL Specification of Management Functions

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

FMT_SMR.1/ADEL Security roles

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

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

FPT_FLS.1/ADEL Failure with preservation of secure state

FPT_FLS.1.1/ADEL The TSF shall preserve a secure state when the following types of failures occur: **the applet deletion manager fails to delete a CAP file/applet as described in [JCRE3], §11.3.4**.
Application note:

- The applet instance deletion must be atomic. The "secure state" referred to in the requirement must comply with the Java Card specifications. That is, if a reset or power fail occurs during the deletion process, then before any applet is selected in card, either the applet instance deletion is completed or the applet shall be selectable and all objects owned by the applet remain unchanged (that is, the functionality of all applet instances on the card remains the same as prior to the unsuccessful deletion attempt) [JCRE3], §11.3.4.

8.1.1.4 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 [JCAPI3].
- 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).

8.1.1.5 CarG Security Functional Requirements

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

FCO_NRO.2/CM Enforced proof of origin

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

Application note:

- Upon reception of a new application CAP file 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 the entity responsible for bytecode verification.

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.

Application note:

- 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 CAP file using an electronic signature mechanism, and no evidence is kept on the card for future verifications.

FCO_NRO.2.3/CM The TSF shall provide a capability to verify the evidence of origin of information to **recipient** given **no limitation**.

FDP_IFC.2/CM Complete information flow control

FDP_IFC.2.1/CM The TSF shall enforce the **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 CAP file by the card through a potentially unsafe communication channel:
- The operations that make information to flow between the subjects are those enabling to send a message through and to receive a message from the communication channel linking the card to the outside world. It is assumed that any message sent through the channel as clear text can be read by the attacker. Moreover, the attacker may capture any message sent through the communication channel and send its own messages to the other subjects.
- The information controlled by the policy is the APDUs exchanged by the subjects through the communication channel linking the card and the CAD. Each of those messages contain part of an application CAP file that is required to be loaded on the card, as well as any control information used by the subjects in the communication protocol.

FDP_IFF.1/CM Simple security attributes

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

Subject / Information	Attribute	value
User	**role**	**Operator, Issuer**
Applet	**checked**	**Boolean**
DAP Key	**OK**	**Boolean**

Application note:

- 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 CAP file has been split into in order to be sent to the card; (3) the ordinal of each piece in the decomposition of the CAP file, etc. See for example Appendix D of [GP23].

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

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

Application note:

- 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 CAP file only if it has received without modification and in the right order all the APDUs sent by the subject S.CAD.

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

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

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

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

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

Application note:

- The verification of the integrity and authenticity evidences can be performed either during loading or during the first installation of an application of the CAP file.

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 be able to **receive** user data in a manner protected from **modification, deletion, insertion, and replay** errors.

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

Application note:

Modification errors should be understood as modification, substitution, unrecoverable ordering change of data and any other integrity error that may cause the application CAP file 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

- JCAPI with already installed applets
- APDUs for Applets

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

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

Application Note:

The list of TSF-mediated actions is implementation-dependent, but CAP file 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 applet **AID** to **None**.

FMT_MSA.3/CM Static attribute initialization

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 **None** to specify alternative initial values to override the default values when an object or information is created.

FMT_SMF.1/CM Specification of Management Functions

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

- The loading of the applet packages, with their AID by the Card Manager.

FMT_SMR.1/CM Security roles

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

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

FTP_ITC.1/CM Inter-TSF trusted channel

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

FTP_ITC.1.2/CM [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 and installing a new application CAP file on the card**.
Application note:

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

8.1.1.6 SCPG Security Functional Requirements

This group contains the security requirements for the smart card platform, that is, operating system and chip that the Java Card System is implemented upon. The requirements are expressed in terms of security functional requirements from [CC2].

FPT_TST.1/SCP TSF Testing

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

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

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

FPT_PHP.3/SCP Resistance to physical attacks

FPT_PHP.3.1/SCP The TSF shall resist [**physical manipulation and physical probing**] to the [**all TOE components implementing the TSF**] by responding automatically such that the SFRs are always enforced.

FPT_RCV.4/SCP Function recovery

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

8.1.1.7 CMGR Group Security Functional Requirements

This group includes requirements for Card Manager.

FDP_ACC.1/CMGR Subset access control

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

FDP_ACF.1/CMGR Security attribute based access control

FDP_ACF.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to objects based on the following:
Subjects: Byte Code Verifier, Operator, Issuer, Card Manager
Objects: applets and keys
Security Attributes: DAP for applets; type and KEK for keys.

FDP_ACF.1.2/CMGR The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:
The Card Manager loads applets into the card on behalf of the Byte Code Verifier.
The Card Manager extradites applets in the card on behalf of the Operator.
The Card Manager locks the loading of applets on the card on behalf of the Issuer.
The Card Manager loads GP keys into the cards on behalf of the Operator.
.
FDP_ACF.1.3/CMGR The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: **none**.

FDP_ACF.1.4/CMGR The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **Only Java packages can be loaded or deleted**.

FMT_MSA.1/CMGR Management of security attributes

FMT_MSA.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to restrict the ability to **modify** the security attributes **code category** to **none**.

FMT_MSA.3/CMGR Static attribute initialization

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

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

8.1.1.8 ASFR Group Security Functional Requirements

This group includes specific requirements for the TOE.

FPT_FLS.1/SpecificAPI Failure with preservation of secure state

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

FPT_ITT.1/SpecificAPI Basic internal TSF data transfer protection

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

FPR_UNO.1/SpecificAPI Unobservability

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

Random Numbers

The TOE generates random numbers. To define the IT security functional requirements of the TOE an additional family (FCS_RNG) of the Class FCS (cryptographic support) is defined in chapter 7.1. This family FCS_RNG Generation of random numbers describes the functional requirements for random number generation used for cryptographic purposes.
The TOE shall meet the requirement “Quality metric for random numbers (FCS_RNG.1)” as specified below (Common Criteria Part 2 extended).

FCS_RNG.1 Random number generation

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

(DRG.4.1) The internal state of the RNG shall use PTRNG of class PTG.2 as random source.
(DRG.4.2) The RNG provides forward secrecy.
(DRG.4.3) The RNG provides backward secrecy even if the current internal state is known.
(DRG.4.4) The RNG provides enhanced forward secrecy after calling the re-seed function that acts as a refreshing done at each random generation.
(DRG.4.5) The internal state of the RNG is seeded by an internal entropy source, PTRNG of class PTG.2.

FCS_RNG.1.2 The TSF shall provide random numbers that meet:

RGS [RGS-B1] and [AIS31] DRG3 & DRG4.(DRG.4.6) The RNG generates output for which 2_{35} strings of bit length 128 are mutually different with probability equal to $(1 – 1/2_{58})$.
(DRG.4.7) Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.

8.1.2 Security Functional Requirements from PACE Module

This section on security functional requirements for the TOE PACE module is divided into sub-section following the main security functionalities.
Operations in this section are in underline font when the SFR’s operation is already present in [PP-EAC2], and in bold font when the operation is done in this ST. When the SFR is refined or assigned in the [PP-EAC2] and additionally refined or assigned in this ST then the font is bold and underline.

Note: actor identifier in the section corresponds to names described in [PP_BAC].

8.1.2.1 Class FCS Cryptographic Support

The TOE shall meet the requirement “Cryptographic key generation (FCS_CKM.1)” as specified below (Common Criteria Part 2). The iterations are caused by different cryptographic key generation algorithms to be implemented and key to be generated by the TOE.

FCS_CKM.1/DH_PACE Cryptographic key generation – Diffie-Hellman for PACE session keys

Hierarchical to:	No other components.
Dependencies:	[FCS_CKM.2 Cryptographic key distribution or FCS_COP.1 Cryptographic operation]: fulfilled by **FCS_COP.1/PACE_ENC**, **FCS_COP.1/PACE_MAC** and **FCS_COP.1/PACE_CAM** FCS_CKM.4 Cryptographic key destruction: fulfilled by **FCS_CKM.4/PACE**
FCS_CKM.1.1 /DH_PACE	The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [selection: *Diffie- Hellman-Protocol compliant to ECDH compliant to [TR-03111]*] and specified cryptographic key sizes **Table 10 column Key size** bit that meet the following: [ICAO-TR-SAC].

Key Usage	algorithm	Key size
/SKPICC-ECDH	**ECDH Key Agreement Algorithm –** [IEEE-P1363]	**160, 192, 224, 256, 320, 384, 512, and 521 bits**
/SKPICC-DH	**DH Key Agreement Algorithm –** [RSA Laboratories, PKCS#3: Diffie-Hellman key-agreement standard, 1993]	**1024/160, 2048/224, 2048/256**
/TDESsession-ECDH	**ECDH Key Agreement Algorithm – 160, 192, 224, 256, 320, 384, 512, and 521 bits**	**112 bits**
/AESsession-ECDH	**ECDH Key Agreement Algorithm – 160, 192, 224, 256, 320, 384, 512, and 521 bits**	**128, 192, 256**
/TDESsession-DH	**DH Key Agreement Algorithm – 1024, 2048 bits MODP Group with 160, 224, 256-bit Prime Order Subgroup**	**112 bits**
/AESsession-DH	**DH Key Agreement Algorithm – 1024, 2048 bits MODP Group with 160, 224, 256-bit Prime Order Subgroup**	**128, 192, 256**

Table 10: FCS_CKM.1/DH_PACE iteration explanation

FCS_CKM.1/PERSO Cryptographic key generation for Session keys

Hierarchical to: No other components
Dependencies: [FCS_CKM.2 Cryptographic key distribution or
FCS_COP.1 Cryptographic operation]: fulfilled by **FCS_COP.1/PERSO**

FCS_CKM.1.1 /PERSO The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm [*assignment: cryptographic key generation algorithm*] and specified cryptographic key sizes [*assignment: cryptographic key sizes*] that meet the following: [*assignment: list of standards*].

Key Usage	algorithm	Key size	standard
/TDES	**TDES ISK key derivation**	**112 bits**	**[ICAO-9303] normative appendix 5**
/GP	**GP session keys**	**112, 128 bits (and 192 & 256 bits for SCP03)**	**[GP211] SCP01, SCP02, or SCP03**

Table 11: FCS_CKM.1/PERSO iteration explanation

The TOE shall meet the requirement “Cryptographic key destruction (FCS_CKM.4)” as specified below (Common Criteria Part 2).

FCS_CKM.4/PACE Cryptographic key destruction

Hierarchical to: No other components
Dependencies: [FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or
FCS_CKM.1 Cryptographic key generation]: fulfilled by **FCS_CKM.1/DH_PACE** and **FCS_CKM.1/PERSO**

FCS_CKM.4.1 PACE The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **Secure erasing of the value by overwriting the data with random numbers** that meets the following: **None**.

FCS_COP.1/PACE_ENC Cryptographic operation – Encryption / Decryption AES / 3DES

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]: fulfilled by **FCS_CKM.1/DH_PACE** FCS_CKM.1/DH_PACEFCS_CKM.4 Cryptographic key destruction: fulfilled by **FCS_CKM.4/PACE**

FCS_COP.1.1 /PACE_ENC: The TSF shall perform secure messaging – encryption and decryption in accordance with a specified cryptographic algorithm **Table 12 algorithm** and cryptographic key sizes **Table 12 Key size** that meet the following: **Table 12 list of standards**.

Algorithm type	algorithm	Key size	List of standards
/ENC_TDES	**TDES in CBC mode**	**112 bits**	**ISO 10116**
/ENC_AES	**AES in CBC mode**	**128, 192, 256**	**ISO 10116**

Table 12: FCS_COP.1/PACE_ENC iteration explanation

FCS_COP.1/PACE_MAC Cryptographic operation – MAC

Hierarchical to: No other components.

Dependencies: [FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation]: fulfilled by **FCS_CKM.1/DH_PACE** FCS_CKM.4 Cryptographic key destruction: fulfilled by **FCS_CKM.4/PACE**

FCS_COP.1.1 /PACE_MAC: The TSF shall perform secure messaging – message authentication code in accordance with a specified cryptographic algorithm **Table 13 algorithm** and cryptographic key sizes **Table 13 Key size** that meet the following: compliant to [ICAO-TR-SAC].

Algorithm explanation	algorithm	Key size	List of standards
/MAC_TDES	**TDES Retail MAC**	**112 bits**	**ISO 9797-1**
/MAC_AES	**AES CMAC**	**128, 192, 256**	**[NIST-800-38B]**

Table 13: FCS_COP.1/PACE_MAC iteration explanation

FCS_COP.1/PACE_CAM Cryptographic operation – Modular Multiplication

Hierarchical to: No other components.

Dependencies: [FCS_CKM.1 Cryptographic key generation]: fulfilled by **FCS_CKM.1/DH_PACE** FCS_CKM.4 Cryptographic key destruction: fulfilled by **FCS_CKM.4/PACE**

FCS_COP.1.1 /PACE_CAM: The TSF shall perform modular multiplication with specify cryptography algorithm and cryptograaokphic key sizes as in Table 14 Key size that meet the following: compliant to: [TR03110-1].

Algorithm type	algorithm	Key size
/CAM_ECDH	**ECC**	**160, 192, 224, 256, 320, 384, 512, 521**

Table 14: FCS_COP.1/PACE_CAM iteration explanation

FCS_COP.1/PERSO Cryptographic operation – Symmetric encryption, decryption, and MAC during manufacturing

Hierarchical to: No other components.
Dependencies: [FDP_ITC.1 Import of user data without security attributes, or
FDP_ITC.2 Import of user data with security attributes, or.
FCS_CKM.1 Cryptographic key generation]: fulfilled by **FCS_CKM.1/PERSO**
FCS_CKM.4 Cryptographic key destruction: fulfilled by **FCS_CKM.4/PACE.**

FCS_COP.1.1 /PERSO: The TSF shall perform **symmetric encryption and decryption** in accordance with a specified cryptographic algorithm **Triple-DES, AES** and cryptographic key sizes **[table 15]** that meet the following: **[table 15]**.

Algorithm type	algorithm	Key size	List of standards
/ENC_TDES	**TDES encryption and decryption**	**112 bits**	**[SP 800-67]**
/ENC_AES	**AES encryption and decryption**	**128, 192, 256**	**[FIPS 197]**
/MAC_TDES	**TDES Retail MAC**	**112 bits**	**ISO 9797-1**
/MAC_AES	**AES CMAC**	**128, 192, 256**	**[NIST-800-38B]**

Table 15: FCS_COP.1/PERSO iteration explanation

FCS_RNG.1/PACE Quality metric for random numbers

Hierarchical to: No other components
Dependencies: No dependencies

FCS_RNG.1.1 /PACE: The TSF shall provide a **hybrid deterministic** random number generator that implements:
DRG.4.1) The internal state of the RNG shall use PTRNG of class PTG.2 as random source.
(DRG.4.2) The RNG provides forward secrecy.
(DRG.4.3) The RNG provides backward secrecy even if the current internal state is known.
(DRG.4.4) The RNG provides enhanced forward secrecy after calling the re-seed function that acts as a refreshing done at each random generation.
(DRG.4.5) The internal state of the RNG is seeded by an internal entropy source, PTRNG of class PTG.2

FCS_RNG.1.2 /PACE: The TSF shall provide random numbers that meet:
RGS [RGS-B1] and [AIS31] DRG3 & DRG4.
(DRG.4.6) The RNG generates output for which 2^35 strings of bit length 128 are mutually different with probability equal to (1 – 1/2^58).
(DRG.4.7) Statistical test suites cannot practically distinguish the random numbers from output sequences of an ideal RNG. The random numbers must pass test procedure A.

Application note: This SFR requires the TOE to generate random numbers used for the authentication protocols as required by FIA_UAU.4.
Regarding the structure of this SFR, even if it is related to the PACE component, the structure comes from [PP-JCSOpen].

8.1.2.2 Class FIA Identification and Authentication

Table 16 provides an overview on the authentication mechanisms used.

Name	SFR for the TOE
Authentication Mechanism for Pre-personalisation Agents	FIA_UAU.1/PERSO FIA_AFL.1/PERSO
Authentication Mechanism for Personalisation Agents	FIA_UAU.4/PACE
Chip Authentication Protocol v.1	FIA_UAU.5/PACE
Terminal Authentication Protocol v.1	FIA_UAU.5/PACE
PACE protocol	*FIA_UAU.1/PACE* *FIA_UAU.5/PACE* *FIA_AFL.1/PACE*
Passive Authentication	FIA_UAU.5/PACE

Table 16: Overview on authentication SFR

FIA_AFL.1/PERSO Authentication failure handling during pre-personalization and personalization phases

Hierarchical to:	No other components.
Dependencies:	FIA_UAU.1 Timing of authentication: fulfilled by **FIA_UAU.1/PERSO**
FIA_AFL.1.1 /Perso	The TSF shall detect when [**Number in Table 17**] unsuccessful authentication attempts occurs related to **authentication attempts** [**defined in Table 17**].
FIA_AFL.1.2 /Perso	When the defined number of unsuccessful authentication attempts has been met, the TSF shall [**Actions in Table 17**].

Auth type	Number	Actions	Authentication attempts from
GP	**3**	**Block GP authentication**.	**GP Authentication key**

Table 17: FIA_AFL.1/PERSO refinements

FIA_AFL.1/PACE Authentication failure handling – PACE authentication using non-blocking authorisation data

Hierarchical to:	No other components.
Dependencies:	FIA_UAU.1 Timing of authentication: fulfilled by **FIA_UAU.1/PACE**
FIA_AFL.1.1 /PACE	The TSF shall detect when [**Number in Table 18**] unsuccessful authentication attempt occurs related to [**Authentication events]**.
FIA_AFL.1.2 /PACE	When the defined number of unsuccessful authentication attempts has been met, the TSF shall [**Actions in Table 18**].

Password	**Number**	**Authentication events**	**Actions**
MRZ, CAN	1	authentication attempts using the PACE password (MRZ, CAN) as shared password	Exponentially increase time delay before new authentication attempt is possible.
PIN & PUK	An administrator configurable positive integer linked to the size of the PIN or PUK (respectively)	Consecutive failed authentication attempts using the PIN or PUK as the shared password for PACE leaving a single authentication attempt	Suspend the PIN or the PUK
	1	On suspend mode, a bad or correct value presentation attempts using the PIN or PUK as the shared password for PACE	Suspend the PIN or the PUK
	1	On suspend mode, After a PACE CAN authentication, a bad PIN/PUK value presentation attempt.	Block the PIN or the PUK

Table 18: FIA_AFL.1/PACE refinements

FIA_UID.1/PERSO Timing of identification

Hierarchical to:	No other components.
Dependencies:	No dependencies.
FIA_UID.1.1 /PERSO	The TSF shall allow **1. to establish a communication channel,** **2. to carry out the mutual authentication Protocol according to [GP]** on behalf of the user to be performed before the user is identified.
FIA_UID.1.2 /PERSO	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/PERSO Timing of authentication

Hierarchical to:	No other components.
Dependencies:	FIA_UID.1 Timing of identification: fulfilled by **FIA_UID.1/PERSO**FIA_UID.1/PERSO
FIA_UAU.1.1 /PERSO	The TSF shall allow **1. to establish a communication channel,** **2. to carry out the mutual authentication Protocol according to [GP]** on behalf of the user to be performed before the user is authenticated.
FIA_UAU.1.2 /PERSO	The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

Application note:

- FIA_AFL.1/PERSO, FIA_UID.1/PERSO, and FIA_UID.1/PERSO are extensions to [PP-EAC2], in order to deal with identification and authentication in pre-personalisation and personalisation phases.

FIA_UID.1/PACE Timing of identification

Hierarchical to:	No other components
Dependencies:	No dependencies
FIA_UID.1.1 /PACE	The TSF shall allow 1. to establish the communication channel, 2. carrying out the PACE Protocol according to [ICAO-TR-SAC], 3. to read the Initialization Data if it is not disabled by TSF according to FMT_MTD.1/INI_DIS 4. to identify themselves by selection of the authentication key. on behalf of the user to be performed before the user is authenticated.
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.

FIA_UAU.1/PACE Timing of authentication

Hierarchical to:	No other components
Dependencies:	FIA_UID.1 Timing of identification: fulfilled by **FIA_UID.1/PACE**
FIA_UAU.1.1 /PACE	The TSF shall allow 1. to establish the communication channel, 2. carrying out the PACE Protocol according to [ICAO-TR-SAC], 3. to read the Initialization Data if it is not disabled by TSF according to FMT_MTD.1/INI_DIS 4. to identify themselves by selection of the authentication key. 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.

FIA_UAU.4/PACE Single-use authentication mechanisms - Single-use authentication of the Terminal by the TOE

Hierarchical to:	No other components
Dependencies:	No dependencies

FIA_UAU.4.1 /PACE	The TSF shall prevent reuse of authentication data related to 1. PACE Protocol according to [ICAO-TR-SAC], 2. Authentication Mechanism based on **Triple-DES, AES** 3. Terminal Authentication Protocol v.1 according to [TR-EAC]

Application note: The authentication mechanisms use a challenge freshly and randomly generated by the TOE to prevent reuse of a response generated by a terminal in a successful authentication attempt.

FIA_UAU.5/PACE Multiple authentication mechanisms

Hierarchical to:	No other components
Dependencies:	No dependencies

FIA_UAU.5.1 /PACE	The TSF shall provide 1. PACE Protocol according to [ICAO-TR-SAC], 2. Secure messaging in MAC-ENC according to [ICAO-TR-SAC], 3. Symmetric Authentication Mechanism based on **Triple-DES, AES** to support user authentication.
FIA_UAU.5.2 /PACE	The TSF shall authenticate any user's claimed identity according to the following rules: **1. TOE accepts the authentication attempt as Pre-personalization Agent by the Symmetric Authentication Mechanism with the Pre-personalization Agent Key.** **2.** 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. 3. The TOE accepts the authentication attempt as Personalization Agent by **the Symmetric Authentication Mechanism with Personalization Agent Key**.

FIA_UAU.6/PACE Re-authenticating – Re-authenticating of Terminal by the TOE

Hierarchical to:	No other components
Dependencies:	No dependencies

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.

8.1.2.3 Class FDP User Data Protection

The TOE shall meet the requirement "Subset access control (FDP_ACC.1)" as specified below (Common Criteria Part 2).

FDP_RIP.1/PACE Subset residual information protection

Hierarchical to:	No other components.
Dependencies:	No dependencies.

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: 1. Session Keys (immediately after closing related communication session). 2. ephemeral private key ephem - $SK_{PICC\ (ECDH/DH)}$ - PACE (by having generated a DH shared secret *K* as defined in [ICAO_TR]). 3. PIN and PUK

8.1.2.4 Class FTP Trusted Path/Channels

FTP_ITC.1/PACE Inter-TSF trusted channel after PACE	
Hierarchical to:	No other components.
Dependencies:	No dependencies.
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	The TSF shall ~~initiate~~ **enforce** communication via the trusted channel for any data exchange between the TOE and the Terminal.

8.1.2.5 Class FMT Security Management

Application note: The SFR FMT_SMF.1 and FMT_SMR.1 provide basic requirements to the management of the TSF data.

The TOE shall meet the requirement “Specification of Management Functions (FMT_SMF.1)” as specified below (Common Criteria Part 2).

FMT_SMF.1/PACE Specification of Management Functions	
Hierarchical to:	No other components
Dependencies:	No dependencies
FMT_SMF.1.1 /PACE	The TSF shall be capable of performing the following management functions: 1. Configuration. 2. Initialize, and resume the PIN or the PUK. 3. Change and unblock the PIN

FMT_SMF.1/PERSO Specification of Management Functions	
Hierarchical to:	No other components
Dependencies:	No dependencies
FMT_SMF.1.1 /PERSO	The TSF shall be capable of performing the following management functions: 1. Initialization , 2. Pre-personalization, 3. Personalization.

The TOE shall meet the requirement “Security roles (FMT_SMR.1)” as specified below (Common Criteria Part 2).

FMT_SMR.1/PACE Security roles

Hierarchical to:	No other components
Dependencies:	FIA_UID.1 Timing of identification fulfilled by **FIA_UID.1/PACE.**
FMT_SMR.1.1 /PACE	The TSF shall maintain the roles 1. Terminal, 2. PACE authenticated BIS-PACE
FMT_SMR.1.2 /PACE	The TSF shall be able to associate users with roles.

FMT_SMR.1/PERSO Security roles

Hierarchical to:	No other components
Dependencies:	FIA_UID.1 Timing of identification fulfilled by **FIA_UID.1/PERSO**.
FMT_SMR.1.1 /PERSO	The TSF shall maintain the roles 3. Manufacturer, 4. Personalization Agent,
FMT_SMR.1.2 /PERSO	The TSF shall be able to associate users with roles.

The TOE shall meet the requirement “Limited capabilities (FMT_LIM.1)” as specified below (Common Criteria Part 2 extended).

FMT_LIM.1/PERSO Limited capabilities

Hierarchical to:	No other components
Dependencies:	FMT_LIM.2 Limited capabilities: fulfilled by **FMT_LIM.2/PERSO**
FMT_LIM.1.1/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.

The TOE shall meet the requirement “Limited availability (FMT_LIM.2)” as specified below (Common Criteria Part 2 extended).

FMT_LIM.2/PERSO Limited availability

Hierarchical to:	No other components
Dependencies:	FMT_LIM.1 Limited capabilities: fulfilled by **FMT_LIM.1/PERSO**
FMT_LIM.2.1/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 of FMT_LIM.1.1 and FMT_LIM.2.1 refers to both IC Dedicated and IC Embedded Software.

The TOE shall meet the requirement “Management of TSF data (FMT_MTD.1)” as specified below (Common Criteria Part 2). The iterations address different management functions and different TSF data.

FMT_MTD.1/INI_ENA Management of TSF data – Writing of Initialization Data and Pre-personalization Data

Hierarchical to:	No other components
Dependencies:	FMT_SMF.1 Specification of management functions: fulfilled by **FMT_SMF.1/PERSO** FMT_SMR.1 Security roles: fulfilled by **FMT_SMR.1/PERSO.**

FMT_MTD.1.1/ INI_ENA	The TSF shall restrict the ability to write the Initialization Data and Pre-personalization Data to the Manufacturer.

Application note: The pre-personalization Data includes but is not limited to the authentication reference data for the Personalization Agent which is the symmetric cryptographic Personalization Agent Key.

FMT_MTD.1/INI_DIS Management of TSF data – Disabling of Read Access to Initialization Data and Pre-personalization Data

Hierarchical to:	No other components
Dependencies:	FMT_SMF.1 Specification of management functions: fulfilled by **FMT_SMF.1/PERSO** FMT_SMR.1 Security roles: fulfilled by **FMT_SMR.1/PERSO.**

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 – Key Read

Hierarchical to:	No other components
Dependencies:	FMT_SMF.1 Specification of management functions: fulfilled by **FMT_SMF.1/PERSO** FMT_SMR.1 Security roles: fulfilled by **FMT_SMR.1/PERSO**.

FMT_MTD.1.1/ KEY_READ	The TSF shall restrict the ability to read the PACE passwords to none.

FMT_MTD.1/Initialize_PINPUK Management of TSF data – Initialize PIN or PUK

Hierarchical to:	No other components
Dependencies:	FMT_SMF.1 Specification of management functions: fulfilled **FMT_SMF.1/PERSO** FMT_SMR.1 Security roles: fulfilled by **FMT_SMR.1/PERSO.**

FMT_MTD.1.1/ Initializer_PINPUK	The TSF shall restrict the ability to write the initial PIN and PUK to the personalization agent.

FMT_MTD.1/Resume_PINPUK Management of TSF data – Resuming PIN or PUK

Hierarchical to:	No other components
Dependencies:	FMT_SMF.1 Specification of management functions: fulfilled by **FMT_SMF.1/PERSO** FMT_SMR.1 Security roles: fulfilled by **FMT_SMR.1/PERSO.**

FMT_MTD.1.1/ Resume_PINPUK	The TSF shall restrict the ability to resume the suspended PIN or the PUK to the eDigitalIdentity document holder.

FMT_MTD.1/Change_PIN Management of TSF data – Changing PIN or PUK

Hierarchical to:	No other components

Dependencies:	FMT_SMF.1 Specification of management functions: fulfilled by **FMT_SMF.1/PERSO** FMT_SMR.1 Security roles: fulfilled by **FMT_SMR.1/PERSO.**
FMT_MTD.1.1/ Change_PIN	The TSF shall restrict the ability to change the PIN to the eDigitalIdentity document holder.

FMT_MTD.1/Unblock_PIN Management of TSF data – Unblocking PIN or PUK

Hierarchical to:	No other components
Dependencies:	FMT_SMF.1 Specification of management functions: fulfilled by **FMT_SMF.1/PERSO** FMT_SMR.1 Security roles: fulfilled by **FMT_SMR.1/PERSO.**
FMT_MTD.1.1/ Unblock_PIN	The TSF shall restrict the ability to unblock the blocked PIN to the eDigitalIdentity document holder (using the PUK for unblocking).

8.1.2.6 Class FPT Protection of the Security Functions

The TOE shall prevent inherent and forced illicit information leakage for User Data and TSF Data. The security functional requirement FPT_EMS.1 addresses the inherent leakage. With respect to the forced leakage they have to be considered in combination with the security functional requirements “Failure with preservation of secure state (FPT_FLS.1)” and “TSF testing (FPT_TST.1)” on the one hand and “Resistance to physical attack (FPT_PHP.3)” on the other. The SFRs “Limited capabilities (FMT_LIM.1)”, “Limited availability (FMT_LIM.2)” and “Resistance to physical attack (FPT_PHP.3)” together with the SAR “Security architecture description” (ADV_ARC.1) prevent bypassing, deactivation and manipulation of the security features or misuse of TOE functions.

The TOE shall meet the requirement “TOE Emanation (FPT_EMS.1)” as specified below (Common Criteria Part 2 extended):

FPT_EMS.1 TOE Emanation

Hierarchical to:	No other components
Dependencies:	No dependencies.
FPT_EMS.1.1	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	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**.

The following security functional requirements address the protection against forced illicit information leakage including physical manipulation.

The TOE shall meet the requirement “Failure with preservation of secure state (FPT_FLS.1)” as specified below (Common Criteria Part 2).

FPT_FLS.1 Failure with preservation of secure state

Hierarchical to: No other components
Dependencies: No dependencies.

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur:
1. Exposure to operating conditions causing a TOE malfunction,
2. failure detected by TSF according to FPT_TST.1.

The TOE shall meet the requirement “TSF testing (FPT_TST.1)” as specified below (Common Criteria Part 2).

FPT_TST.1 TSF testing

Hierarchical to: No other components
Dependencies: No dependencies.

FPT_TST.1.1 The TSF shall run a suite of self-tests **[see Table 19: FPT_TST triggering conditions]** 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.

Conditions under which self-test should occur	Description of the self-test
During initial start-up	**RNG live test, sensor test, FA detection, Integrity Check of NVM ES**
Periodically	**RNG monitoring, FA detection**
After cryptographic computation	**FA detection**
Before any use or update of TSF data	**FA detection, Integrity Check of related TSF data**

Table 19: FPT_TST triggering conditions

The TOE shall meet the requirement “Resistance to physical attack (FPT_PHP.3)” as specified below (Common Criteria Part 2).

FPT_PHP.3 Resistance to physical attack

Hierarchical to: No other components
Dependencies: No dependencies.

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

8.1.3 Security Functional Requirements for Patch Management

FMT_SMR.1/OS-UPDATE Security roles

FMT_SMR.1.1/OS-UPDATE The TSF shall maintain the roles **OS Developer, OS Patch Loader, Issuer**.

FMT_SMR.1.2/OS-UPDATE The TSF shall be able to associate users with roles.

FMT_SMF.1/OS-UPDATE Specification of Management Functions

FMT_SMF.1.1/OS-UPDATE The TSF shall be capable of performing the following management functions: **activation of additional code.**

Application Note:
Once verified and installed, additional code is become immediately effective.

FIA_ATD.1/OS-UPDATE User attribute definition

FIA_ATD.1.1/OS-UPDATE The TSF shall maintain the following list of security attributes belonging to individual users: **additional code ID for each activated additional code**.

Refinement: "Individual users" stands for additional code.

FDP_ACC.1/OS-UPDATE Subset access control

FDP_ACC.1.1/OS-UPDATE The TSF shall enforce the **OS Update Access Control Policy** on **the following list of subjects, objects and operations:**

- **Subjects: S.OS-Developer is the representative of the OS Developer within the TOE, who responsible for verifying the signature and decrypting the additional code before authorizing its loading, installation and activation, [None]**
- **Objects: additional code and associated cryptographic signature**
- **Operations: loading, installation and activation of additional code**

FDP_ACF.1/OS-UPDATE Security attribute based access control

FDP_ACF.1.1/OS-UPDATE The TSF shall enforce the **OS Update Access Control Policy** to objects based on the following:

- **Security Attributes:**
  - **The additional code cryptographic signature verification status**
  - **The Identification Data verification status (between the Initial TOE and the additional code)**

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

- **The verification of the additional code cryptographic signature (using D.OS-UPDATE_SGNVER-KEY) by S.OS-Developer is successful.**
- **The decryption of the additional code prior installation (using D.OS-UPDATE_DEC-KEY) by S.OS-Developer is successful.**
- **The comparison between the identification data of both the Initial TOE and the additional code demonstrates that the OS Update operation can be performed.**
- **[None]**

FDP_ACF.1.3/OS-UPDATE The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **[None]**.

FDP_ACF.1.4/OS-UPDATE The TSF shall explicitly deny access of subjects to objects based on the following additional rules: **[None].**

Application Note:
Identification data verification is necessary to ensure that the received additional code is actually targeting the TOE and that its version is compatible with the TOE version.

Confidentiality protection must be enforced when the additional code is transmitted to the TOE for loading (See OE.OS-UPDATE-ENCRYPTION). Confidentiality protection can be achieved either through direct encryption of the additional code, or by means of a trusted path ensuring the confidentiality of the communication to the TOE.

FMT_MSA.3/OS-UPDATE Security attribute initialisation

FMT_MSA.3.1/OS-UPDATE The TSF shall enforce the **OS Update Access Control Policy** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/OS-UPDATE The TSF shall allow the **OS Developer** to specify alternative initial values to override the default values when an object or information is created.

Application Note:

The additional code signature verification status must be set to “Fail” by default, therefore preventing any additional code from being installed until the additional code signature is actually successfully verified by the TOE.

FTP_TRP.1/OS-UPDATE Trusted Path

FTP_TRP.1.1/OS-UPDATE The TSF shall provide a communication path between itself and **remote** that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from **[none]**.

FTP_TRP.1.2/OS-UPDATE The TSF shall permit **remote users** to initiate communication via the trusted path.

FTP_TRP.1.3/OS-UPDATE The TSF shall require the use of the trusted path for **the transfer of the additional code to the TOE.**

Application Note:

During the transmission of the additional code to the TOE for loading the confidentiality shall be ensured either through direct encryption of the additional code, or by means of a trusted path ensuring the confidentiality of the communication to the TOE.

In case that the additional code is encrypted independently of the trusted path the ST writer can select ‘none’ in FTP_TRP.1.1/OS-UPDATE.

Otherwise, the trusted path shall ensure the confidentiality of the transmitted additional code. In this case the ST writer shall select ‘disclosure’ in FTP_TRP.1.1/OS-UPDATE.

FCS_COP.1/OS-UPDATE-DEC Cryptographic operation

FCS_COP.1.1/OS-UPDATE-DEC The TSF shall perform **Decryption of the additional code prior installation** in accordance with a specified cryptographic algorithm **[AES-CBC]** and cryptographic key sizes **[AES-256]** that meet the following: **[assignment:** AES-CBC ISO9797-M2 NIST SP800-38A**]**.

FCS_COP.1/OS-UPDATE-VER Cryptographic operation

FCS_COP.1.1/OS-UPDATE-VER The TSF shall perform **digital signature verification of the additional code to be loaded** in accordance with a specified cryptographic algorithm **[AES-CMAC]** and cryptographic key sizes **[AES-256]** that meet the following: **[assignment:** NIST SP800-38B**]**.

FPT_FLS.1/OS-UPDATE Failure with preservation of secure state

FPT_FLS.1.1/OS-UPDATE The TSF shall preserve a secure state when the following types of failures occur: **interruption or incident which prevents the forming of the Updated TOE**.

Application Note:

The OS Update operation must be either successful, or fail securely. The TOE code and identification data must be updated in an atomic way in order to always be consistent. In case of interruption or incident during the OS Update operation, the OS Developer may choose to implement any technical behavior, provided that the TOE remains in a secure state, for example by canceling the operation (the TOE remains the Initial TOE) or entering an error state, and consistency is maintained between the TOE code and the ID data.

The ST writer shall describe the “secure state” to which the OS update might lead.

8.2 SECURITY ASSURANCE REQUIREMENTS

The security assurance requirement level is EAL6 augmented with ALC_FLR.1 .
The list of all the security assurance requirements for this security target is defined in the Table 20: Assurance Level 6 (EAL6)".
The entry "EAL6" means that this requirement is defined in the CC part 5
The entry "EAL6/PP" means that requirement is defined in both [CC-3] part and in [PP-JCS-Open] (or linked)
The entry "ST" means that the requirement is defined in this security target.

SAR	Title	Required by
ADV: Development	ADV_ARC.1 Security architecture description	EAL6 / PP
	ADV_FSP.5 Complete semi-formal functional specification with additional error information	EAL6
	ADV_IMP.2 Complete mapping of the implementation representation of the TSF	EAL6
	ADV_INT.3 Minimally complex internals	EAL6
	ADV_SPM.1 Formal TOE security policy model	EAL6
	ADV_TDS.5 Complete semiformal modular design	EAL6
AGD: Guidance documents	AGD_OPE.1 Operational user guidance	EAL6 / PP
	AGD_PRE.1 Preparative procedures	EAL6 / PP
ALC: Life-cycle support	ALC_CMC.5 Advanced support	EAL6
	ALC_CMS.5 Development tools CM coverage	EAL6
	ALC_DEL.1 Delivery procedures	EAL6 / PP
	ALC_DVS.2 Sufficiency of security measures	EAL6
	ALC_LCD.1 Developer defined life-cycle model	EAL6 / PP
	ALC_TAT.3 Compliance with implementation standards - all parts	EAL6
	ALC_FLR.1 Basic flaw remediation	**ST**
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims	EAL6 / PP
	ASE_ECD.1 Extended components definition	EAL6 / PP
	ASE_INT.1 ST introduction	EAL6 / PP
	ASE_OBJ.2 Security objectives	EAL6 / PP
	ASE_REQ.2 Derived security requirements	EAL6 / PP
	ASE_SPD.1 Security problem definition	EAL6 / PP
	ASE_TSS.1 TOE summary specification	**ST**
ATE: Tests	ATE_COV.3 Rigorous analysis of coverage	EAL6
	ATE_DPT.3 Testing: modular design	EAL6
	ATE_FUN.2 Ordered functional testing	EAL6
	ATE_IND.2 Independent testing - sample	EAL6 / PP
AVA: Vulnerability assessment	AVA_VAN.5 Advanced methodical vulnerability analysis	EAL6

Table 20: Assurance Level 6 (EAL6)

Among the set of assurance components chosen for EAL6, the assignment appears only in ADV_SPM.1. The assignment used in ADV_SPM.1 is defined as follows:

ADV_SPM.1 **Formal TOE security policy model**
Dependencies: ADV_FSP.4

Developer action elements:

ADV_SPM.1.1D The developer shall provide a formal security policy model for the **Virtual Machine Access Policy**:

- Access Control Policy: FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL
- Flow control: FDP_IFC.1/JCVM, FDP_IFF.1/JCVM
- Security Attributes: FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_MSA.2/FIREWALL_JCVM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM
- Security roles: FMT_SMR.1/JCRE
- Management Functions: FMT_SMF.1/CORE_LC
- TSF Data: FMT_MTD.1/JCRE

Note: For this formal modelisation, we focus on JCVM opcode processing. The Applet Install, Delete and APIs are out the scope of this modelisation. The initial settings (the Selected Applet Context and the initial active applet) are also out of the scope because done before the JCVM entering (selection of the applet)

Note: For this formal modelisation, the SPM scope will be considering one VM execution

ADV_SPM.1.2D For each policy covered by the formal security policy model, the model shall identify the relevant portions of the statement of SFRs that make up that policy.

ADV_SPM.1.3D The developer shall provide a formal proof of correspondence between the model and any formal functional specification.

ADV_SPM.1.4D The developer shall provide a demonstration of correspondence between the model and the functional specification.

The SFR FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FMT_SMF.1/ADEL, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_SMF.1/CM, FDP_ITC.2/Installer, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMF.1, are out of the scope of the SPM as they are linked to the applet loading or deletion that is out of scope of the SPM boundaries limited to VM opcodes

The SFR FMT_MTD.3/JCRE is out of scope of the SPM modelisation because AID registry is created during loading phase, which is also out of scope of the SPM (Hypothesis 2 of the SPM document [SPM]).

8.3 SECURITY REQUIREMENTS RATIONALE

8.3.1 OBJECTIVES for PP JCS – OPEN Configuration

	O.SID	O.OPERATE	O.RESOURCES	O.FIREWALL	O.NATIVE	O.REALLOCATION	O.GLOBAL_ARRAYS_CONFID	O.GLOBAL_ARRAYS_INTEG	O.ARRAY_VIEWS_CONFID	O.ARRAY_VIEWS_INTEG	O.ALARM	O.TRANSACTION	O.CIPHER	O.RNG	O.PIN_MNGT	O.KEY_MNGT	O.OBJ_DELETION	O.INSTALL	O.LOAD	O.DELETION	O.SCP.RECOVERY	O.SCP.SUPPORT	O.SCP.IC	O.CARD-MANAGEMENT	O.SpecificAPI
FDP_IFC.1/JCVM				x			x	x	x	x															
FDP_IFF.1/JCVM				x			x	x	x	x															
FDP_RIP.1/OBJECTS						x	x					x			x	x									
FMT_MSA.2/FIREWALL_JCVM				x																					
FMT_MSA.3/FIREWALL	x			x																					
FMT_MSA.3/JCVM	x			x																					
FMT_SMR.1/JCRE			x	x																					
FMT_SMF.1/CORE_LC			x	x																					
FCS_CKM.1													x			x									
FCS_CKM.2													x			x									
FCS_CKM.3													x			x									
FCS_CKM.4													x			x									
FCS_COP.1													x			x									
FDP_RIP.1/APDU						x	x					x			x	x									
FDP_RIP.1/bArray						x	x		x			x			x	x									
FDP_RIP.1/GlobalArray						x	x					x			x	x									
FDP_RIP.1/ABORT						x	x					x			x	x									
FDP_RIP.1/KEYS						x	x					x			x	x									
FDP_ROL.1/FIREWALL		x	x									x			x										
FAU_ARP.1		x	x								x														
FDP_SDI.2/DATA															x	x									
FPT_TDC.1		x																							
FPT_FLS.1/JCS		x	x								x														
FPR_UNO.1													x		x	x									
FMT_MTD.1/JCRE	x		x	x																					
FMT_MTD.3/JCRE	x		x	x																					
FIA_ATD.1/AID	x	x																							
FIA_UID.2/AID	x																								
FIA_USB.1/AID	x	x																							
FDP_ITC.2/Installer	x	x		x														x							

	O.SID	O.OPERATE	O.RESOURCES	O.FIREWALL	O.NATIVE	O.REALLOCATION	O.GLOBAL_ARRAYS_CONFID	O.GLOBAL_ARRAYS_INTEG	O.ARRAY_VIEWS_CONFID	O.ARRAY_VIEWS_INTEG	O.ALARM	O.TRANSACTION	O.CIPHER	O.RNG	O.PIN_MNGT	O.KEY_MNGT	O.OBJ_DELETION	O.INSTALL	O.LOAD	O.DELETION	O.SCP.RECOVERY	O.SCP.SUPPORT	O.SCP.IC	O.CARD-MANAGEMENT	O.SpecificAPI
FMT_SMR.1/Installer			X	X																					
FPT_FLS.1/Installer		X	X								X							X							
FPT_RCV.3/Installer		X	X															X		X					
FMT_MSA.1/ADEL	X			X																X					
FMT_MSA.3/ADEL	X			X																X					
FMT_SMR.1/ADEL			X	X																X					
FMT_SMF.1/ADEL	X		X	X																					
FDP_ACC.2/ADEL																				X					
FDP_ACF.1/ADEL																				X					
FDP_RIP.1/ADEL						X	X					X			X	X				X					
FPT_FLS.1/ADEL		X	X								X									X					
FDP_ACC.2/FIREWALL		X		X											X										
FDP_ACF.1/FIREWALL		X		X	X										X										
FMT_MSA.1/JCRE	X			X																					
FMT_MSA.1/JCVM	X			X																					
FDP_RIP.1/TRANSIENT						X	X					X			X	X									
FDP_RIP.1/ODEL						X	X					X			X	X	X								
FPT_FLS.1/ODEL		X	X								X						X								
FMT_MSA.1/CM	X			X																					
FMT_MSA.3/CM	X			X																					
FMT_SMR.1/CM			X	X																					
FMT_SMF.1/CM	X		X	X																				X	
FCO_NRO.2/CM																			X						
FIA_UID.1/CM																			X						
FDP_IFC.2/CM																			X						
FDP_IFF.1/CM																			X						
FDP_UIT.1/CM																			X						
FTP_ITC.1/CM																			X						
FPT_TST.1/SCP																						X			
FPT_PHP.3/SCP																							X		
FPT_RCV.4/SCP																					X				
FDP_ACC.1/CMGR																								X	
FDP_ACF.1/CMGR																								X	
FMT_MSA.1/CMGR																								X	
FMT_MSA.3/CMGR																								X	

	O.SID	O.OPERATE	O.RESOURCES	O.FIREWALL	O.NATIVE	O.REALLOCATION	O.GLOBAL_ARRAYS_CONFID	O.GLOBAL_ARRAYS_INTEG	O.ARRAY_VIEWS_CONFID	O.ARRAY_VIEWS_INTEG	O.ALARM	O.TRANSACTION	O.CIPHER	O.RNG	O.PIN_MNGT	O.KEY_MNGT	O.OBJ_DELETION	O.INSTALL	O.LOAD	O.DELETION	O.SCP.RECOVERY	O.SCP.SUPPORT	O.SCP.IC	O.CARD-MANAGEMENT	O.SpecificAPI
FPT_FLS.1/SpecificAPI																									X
FPT_ITT.1/SpecificAPI																									X
FPR_UNO.1/SpecificAPI																									X
FCS_RNG.1														X											

Table 21: rationale objective vs. SFR

8.3.1.1 SECURITY OBJECTIVES FOR THE TOE

8.3.1.1.1 IDENTIFICATION

O.SID Subjects' identity is AID-based (applets, CAP files), 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.
Installation procedures ensure protection against forgery (the AID of an applet is under the control of the TSFs) or re-use of identities (FIA_UID.2/AID, FIA_USB.1/AID).

8.3.1.1.2 EXECUTION

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

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

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

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

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

O.GLOBAL_ARRAYS_CONFID Only arrays can be designated as global, and the only global arrays required in the Java Card API are the APDU buffer, 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/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.

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

O.GLOBAL_ARRAYS_INTEG This objective is met by the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM), which prevents an application from keeping a pointer to the APDU buffer of the card, 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.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.

The confidentiality of the residual information of the array is ensured by FDP_RIP.1/bArray.

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.

8.3.1.1.3 SERVICES

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

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).

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

O.RNG This security objective is directly covered by FCS_RNG.1 which ensures the cryptographic quality of random number generation.

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.

O.KEY-MNGT This relies on the same security functional requirements as O.CIPHER, plus FDP_RIP.1 and FDP_SDI.2/DATA as well. Precisely it is met by the following components: FCS_CKM.1, ***FCS_CKM.2,*** ***FCS_CKM.3,*** FCS_CMK.4, FCS_COP.1, FPR_UNO.1, 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.

8.3.1.1.4 OBJECT DELETION

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

8.3.1.1.5 APPLET MANAGEMENT

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

O.LOAD This security objective specifies that the loading of a CAP file into the card must be secure. Evidence of the origin of the CAP file 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.DELETION This security objective specifies that applet and CAP file 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 CAP file is a by-product of this policy as well. Non-accessibility of deleted data is met by FDP_RIP.1/ADEL and the TSFs are protected against possible failures of the deletion procedures (FPT_FLS.1/ADEL, FPT_RCV.3/Installer). The security functional requirements of the class FMT (FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, and FMT_SMR.1/ADEL) included in the group ADELG also contribute to meet this objective.

8.3.1.1.6 SCP

O.SCP.RECOVERY This security objective specifies that the platform must behave securely if an unexpected loss of power occurs. This is covered by FPT_RCV.4/SCP which specifies the recovery after unexpected power failure.

O.SCP.SUPPORT This security objective specifies that the SCP provides security features to the JCS. This is provided by FPT_TST.1/SCP. This is also provided by requirements of the IC, which are described in [IFX-IC].

O.SCP.IC This security objective specifies that the IC must provide mechanisms to protect itself against physical attacks. This is provided by FPT_PHP.3/SCP. This is also provided by requirements of the IC, which are described in [IFX-IC].

8.3.1.1.7 Card Management

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

8.3.1.1.8 ASFR

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

O.RNG The security objective O.RNG is met by the following SFR FCS_RNG.1/PACE.

8.3.2 Security Functional Requirements Rationale for PACE Module

The rationale in this paragraph comes from [PP- EAC2] §6.3.1

	OT.AC_Pers	OT.Data_Integrity	OT.Data_Authenticity	OT.Data_Confidentiality	OT.Identification	OT.Prot_Abuse_Func	OT.Prot_Inf_Leak	OT.Prot_Phys_Tamper	OT.Prot_Malfunction
FCS_CKM.1/DH_PACE (o)	X	X	X	X					
FCS_CKM.1/PERSO (p)	X	X	X	X					
FCS_CKM.4/PACE (o)(p)	X	X	X	X					
FCS_COP.1/PACE_ENC (o)				X					
FCS_COP.1/PACE_MAC (o)	X	X	X						
FCS_COP.1/PACE_CAM (o)					X				
FCS_COP.1/PERSO (p)	X	X	X	X					
FCS_RNG.1/PACE (o) (p)	X		X	X					
FDP_RIP.1/PACE(p)	X	X	X	X					
FIA_AFL.1/PERSO (p)	X	X	X	X					
FIA_AFL.1/PACE (o)		X	X	X					
FIA_UID.1/PERSO (p)	X	X	X	X					
FIA_UAU.1/PERSO (p)	X	X	X	X					
FIA_UID.1/PACE (o)	X	X	X	X					
FIA_UAU.1/PACE (o)	X	X	X	X					
FIA_UAU.4/PACE (o)	X	X	X	X					
FIA_UAU.5/PACE (o)	X	X	X	X					
FIA_UAU.6/PACE (o)	X	X	X	X					
FTP_ITC.1/PACE (o)	X	X	X	X					
FMT_SMF.1/PACE (o)		X	X	X	X				
FMT_SMF.1/PERSO (p)	X	X	X	X	X				
FMT_SMR.1/PACE (o)		X	X	X	X				
FMT_SMR.1/PERSO (o)	X	X	X	X	X				
FMT_LIM.1/PERSO (o) (p)						X			
FMT_LIM.2/PERSO (o) (p)						X			
FMT_MTD.1/INI_ENA (p)					X				
FMT_MTD.1/INI_DIS (p)					X				
FMT_MTD.1/KEY_READ (o)	X			X					
FMT_MTD.1/Initialize_PINPUK(p)		X	X	X					
FMT_MTD.1/Resume_PINPUK(o) (p)		X	X	X					
FMT_MTD.1/Change_PIN(o) (p)		X	X	X					
FMT_MTD.1/Unblock_PIN(o) (p)		X	X	X					
FPT_EMS.1 (o) (p)							X		
FPT_FLS.1 (o) (p)							X		X
FPT_TST.1 (o) (p)							X		X
FPT_PHP.3 (o) (p)	X	X		X			X	X	

Table 22: Security Functional Requirement Rationale

Note: SFR followed by (o) (respectively (p)) means SFR is applicable in Operational phase (respectively (p)) personalization phase.

The security objective **OT.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. The SFR FMT_MTD.1/INI_ENA and FCS_COP.1/PACE_CAM allow 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, FMT_SMF.1/PERSO and FMT_SMR.1/PACE support the functions and roles related.

the card issuing life cycle phases of the application data requiring PACE usage

The security objective **OT.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 with PACE authentication using FCS_CKM.1/DH_PACE,.The SFR FCS_RNG.1/PACE represents a general support for cryptographic operations needed. In pre-personalisation, the SFR FCS_CKM.1/PERSO and FCS_COP.1/PERSO ensure the integrity of data transfers after successful authentication of the pre-personalisation agent according to FIA_UID.1/PERSO and FIA_UAU.1/PERSO with the support of FIA_AFL.1/PERSO. The FDP_RIP.1/PACE require erasing the values of session keys.
the TSF data are protected in confidentiality and integrity against physical manipulation by FPT_PHP.3. The FDP_RIP.1/PACE requires erasing the values of session keys.
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, FIA_UAU.6/PACE and FCS_CKM.4/PACE represent some required specific properties of the protocols used.
Unauthorised modifying of the exchanged data is addressed, in the first line, by FTP_ITC.1/PACE using FCS_COP.1/PACE_MAC.
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/PERSO manages the roles (including Personalization Agent) and the SFR FMT_SMF.1/PERSO lists the TSF management functions (including Personalization).

The security objective **OT.Data_Integrity** “Application data” requires the TOE to protect the integrity of the application data requiring usage of PACE (e.g. logical travel document) stored on the TOE against physical manipulation and unauthorized writing. Physical manipulation is addressed by FPT_PHP.3.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/PACE represent some required specific properties of the protocols used. The SFR FMT_SMR.1/PACE & FMT_SMR.1/PERSO manage the roles and the SFR FMT_SMF.1/PACE & FMT_SMF.1/PERSO manage 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 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 K_{MAC}).
The session keys are destroyed according to FCS_CKM.4/PACE after use.
In pre-personalisation, the SFR FCS_CKM.1/PERSO and FCS_COP.1/PERSO ensure the integrity of data transfers after successful authentication of the pre-personalisation agent according to FIA_UID.1/PERSO and FIA_UAU.1/PERSO, with the support of FIA_AFL.1/PERSO.

The security objective **OT.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 or Chip and Terminal Authentication v.1 (FIA_UID.1/PACE, FIA_UAU.1/PACE) using FCS_CKM.1/DH_PACE and possessing the special properties FIA_UAU.5/PACE, FIA_UAU.6/PACE. FDP_RIP.1/PACE requires erasing the values of session keys (here: for KMAC). FIA_UAU.4/PACE, FIA_UAU.5/PACE and FCS_CKM.4/PACE represent some required specific properties of the protocols used. 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.The SFR FCS_RNG.1/PACE represents a general support for cryptographic operations needed. The SFR FMT_SMR.1/PACE & FMT_SMR.1/PERSO manage the roles and the SFR FMT_SMF.1/PACE & FMT_SMF.1/PERSO manage the TSF management functions.

In pre-personalisation, the SFR FCS_CKM.1/PERSO and FCS_COP.1/PERSO ensure the authenticity of data transfers after successful authentication of the pre-personalisation agent according to FIA_UID.1/PERSO and FIA_UAU.1/PERSO, with the support of FIA_AFL.1/PERSO.

The security objective **OT.Data_Confidentiality** aims that the TOE always ensures confidentiality of the User and TSF data stored and, after the PACE Physical manipulation is addressed by FPT_PHP.3. FIA_UAU.4/PACE, FIA_UAU.5/PACE and FCS_CKM.4/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 and possessing the special properties FIA_UAU.5/PACE, FIA_UAU.6/PACE. FDP_RIP.1/PACE requires erasing the values of session keys (here: for K_{ENC}). 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.The SFR FCS_RNG.1/PACE represents the general support for cryptographic operations needed. The SFR FMT_SMR.1/PACE, & FMT_SMR.1/PERSO manage the roles and the SFR FMT_SMF.1/PACE & FMT_SMF.1/PERSO manage the TSF management functions. In pre-personalisation, the SFR FCS_CKM.1/PERSO and FCS_COP.1/PERSO ensure the confidentiality of data transfers after successful authentication of the pre-personalisation agent according to FIA_UID.1/PERSO and FIA_UAU.1/PERSO, with the support of FIA_AFL.1/PERSO.
The SFR FMT_MTD.1/KEY_READ requires that data cannot be unauthorized read afterwards.
The security objective **OT.Prot_Abuse_Func** “Protection against Abuse of Functionality” is ensured by the SFR FMT_LIM.1/PERSO and FMT_LIM.2/PERSO which prevent misuse of test functionality of the TOE or other features which may not be used after TOE Delivery.

The security objective **OT.Prot_Inf_Leak** “Protection against Information Leakage” requires the TOE to protect confidential TSF data stored and/or processed in the travel 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,
- 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.

The security objective **OT.Prot_Phys_Tamper** “Protection against Physical Tampering” is covered by the SFR FPT_PHP.3.

The security objective **OT.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.

Additionally to OT.Data_Integrity, OT.Data_Authenticity and **OT.Data_Confidentiality**

Since PACE can use the PIN as the shared secret, using and management of PIN, the SFRs **FIA_AFL.1/PACE, FMT_MTD.1/Initialize_PINPUK, FMT_MTD.1/Resume_PINPUK, FMT_MTD.1/Change_PIN, FMT_MTD.1/Unblock_PIN** support the achievement of these objectives.

8.3.3 Security Functional Requirements Rationale for Patch Management

	O.SECURE_LOAD_ACODE	O.SECURE_AC_ACTIVATION	O.TOE_IDENTIFICATION	O.CONFID-OS-UPDATE.LOAD
FDP_ACC.1/OS-UPDATE	X	X	X	X
FDP_ACF.1/OS-UPDATE	X	X	X	X
FIA_ATD.1/OS-UPDATE			X	
FMT_MSA.3/OS-UPDATE	X	X	X	X
FMT_SMR.1/OS-UPDATE	X	X	X	X
FMT_SMF.1/OS-UPDATE	X	X	X	X
FTP_TRP.1/OS-UPDATE				X
FCS_COP.1/OS-UPDATE-DEC				X
FCS_COP.1/OS-UPDATE-VER	X			
FPT_FLS.1/OS-UPDATE	X	X	X	

Table 23: Security Functional Requirement Rationale for Patch Management

O.SECURE_LOAD_ACODE This security objective specifies that the TOE shall check the authenticity and the integrity of the additional code to be loaded. This is covered by FDP_ACC.1/OS-UPDATE, FDP_ACF.1/OS-UPDATE, FMT_MSA.3/OS-UPDATE, FMT_SMR.1/OS-UPDATE, FMT_SMF.1/OS-UPDATE, SFR FCS_COP.1/OS-UPDATE-VER that define the different access control policies for the authenticity and the integrity.
Any interruption or incident will prevent the forming and activation of the additional code. It is covered by the FPT_FLS.1/OS-UPDATE.

O.SECURE_AC_ACTIVATION This security objective specifies that the activation of the additional code and update of the Identification Data shall be performed at the same time in an atomic way. This is covered by FDP_ACC.1/OS-UPDATE, FDP_ACF.1/OS-UPDATE, FMT_MSA.3/OS-UPDATE, FMT_SMR.1/OS-UPDATE, FMT_SMF.1/OS-UPDATE that define the different access control policies.
Any interruption or incident will prevent the forming and activation of the additional code. It is covered by the FPT_FLS.1/OS-UPDATE.

O.TOE_IDENTIFICATION This security objective specifies the identifications of both the Initial TOE and additional code. This is covered by FDP_ACC.1/OS-UPDATE , FDP_ACF.1/OS-UPDATE, FIA_ATD.1/OS-UPDATE, FIA_ATD.1/OS-UPDATE , FMT_MSA.3/OS-UPDATE, FMT_SMR.1/OS-UPDATE, FMT_SMF.1/OS-UPDATE
Any interruption or incident will prevent any change of the identification data. It is covered by the FPT_FLS.1/OS-UPDATE.

O.CONFID-OS-UPDATE.LOAD This security objective specifies that The TOE shall decrypt the additional code prior installation. This is covered by FDP_ACC.1/OS-UPDATE, FDP_ACF.1/OS-UPDATE, FMT_MSA.3/OS-UPDATE, FMT_SMR.1/OS-UPDATE, FMT_SMF.1/OS-UPDATE, FTP_TRP.1/OS-UPDATE, FCS_COP.1/OS-UPDATE-DEC to cover this confidentiality objective.

8.3.4 DEPENDENCIES for PP JCS-OPEN CONFIGURATION

8.3.4.1 SFRS DEPENDENCIES

Requirements	CC dependencies	Satisfied dependencies
FAU_ARP.1	FAU_SAA.1	Unsupported
FCO_NRO.2/CM	FIA_UID.1	FIA_UID.1/CM
FCS_CKM.1	(FCS_CKM.2 or FCS_COP.1), FCS_CKM.4	FCS_CKM.2, FCS_CKM.4
FCS_CKM.2	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1, FCS_CKM.4
FCS_CKM.3	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1, FCS_CKM.4
FCS_CKM.4	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2	FCS_CKM.1,
FCS_COP.1	(FCS_CKM.1 or FDP_ITC.1 or FDP_ITC.2), FCS_CKM.4	FCS_CKM.1, FCS_CKM.4
FDP_ACC.2/ADEL	FDP_ACF.1	FDP_ACF.1/ADEL
FDP_ACC.2/FIREWALL	FDP_ACF.1	FDP_ACF.1/FIREWALL
FDP_ACF.1/ADEL	FDP_ACC.1, FMT_MSA.3	FDP_ACC.2/ADEL , FMT_MSA.3/ADEL
FDP_ACF.1/FIREWALL	FDP_ACC.1, FMT_MSA.3	FDP_ACC.2/FIREWALL, FMT_MSA.3/FIREWALL
FDP_IFC.1/JCVM	FDP_IFF.1	FDP_IFF.1/JCVM
FDP_IFC.2/CM	FDP_IFF.1	FDP_IFF.1/CM
FDP_IFF.1/JCVM	FDP_IFC.1, FMT_MSA.3	FDP_IFC.1/JCVM, FMT_MSA.3/JCVM
FDP_IFF.1/CM	FDP_IFC.1, FMT_MSA.3	FDP_IFC.2/CM, FMT_MSA.3/CM
FDP_ITC.2/Installer	(FDP_ACC.1 or FDP_IFC.1), FPT_TDC.1, (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM, FPT_TDC.1
FDP_RIP.1/OBJECTS	none	
FDP_RIP.1/APDU	none	
FDP_RIP.1/bArray	none	
FDP_RIP.1/ABORT	none	
FDP_RIP.1/KEYS	none	
FDP_RIP.1/ADEL	none	
FDP_RIP.1/TRANSIENT	none	
FDP_RIP.1/ODEL	none	
FDP_RIP.1/GlobalArray	none	
FDP_ROL.1/FIREWALL	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM
FDP_SDI.2/DATA	none	
FIA_ATD.1/AID	none	
FIA_UID.1/CM	none	
FIA_UID.2/AID	none	
FDP_UIT.1/CM	(FDP_ACC.1 or FDP_IFC.1), (FTP_ITC.1 or FTP_TRP.1)	FDP_IFC.2/CM, FTP_ITC.1/CM
FIA_USB.1/AID	FIA_ATD.1	FIA_ATD.1/AID
FMT_MSA.1/ADEL	(FDP_ACC.1 or FDP_IFC.1), FMT_SMF.1, FMT_SMR.1	FDP_ACC.2/ADEL, FMT_SMF.1/ADEL, FMT_SMR.1/ADEL

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

Requirements	CC dependencies	Satisfied dependencies
FPR_UNO.1/SpecificAPI.	none	
FCS_RNG.1	none	

Table 24: SFR dependencies

8.3.4.1.1 RATIONALE FOR THE EXCLUSION OF DEPENDENCIES

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

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

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

8.3.5 DEPENDENCIES for PACE Module

The rationale in this paragraph comes from [PP_EAC2] §6.3.2.

SFR	Dependencies	Support of the dependencies
FCS_CKM.1/DH_PACE	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4	**FCS_COP.1/PACE_ENC,** **FCS_COP.1/PACE_CAM** **FCS_COP1/PACE_MAC** **FCS_CKM.4/PACE**
FCS_CKM.1/PERSO	[FCS_CKM.2 or FCS_COP.1], FCS_CKM.4	**FCS_COP.1/PERSO** **NA: Perso Keys are not erased in Perso**
FCS_CKM.4/PACE	[FDP_ITC.1 or FDP_ITC.2, or FCS_CKM.1]	**FCS_CKM.1/DH_PACE,** **FCS_CKM.1/PERSO**
FCS_COP.1/PACE_ENC	[FDP_ITC.1 or FDP_ITC.2, or FCS_CKM.1], FCS_CKM.4	**FCS_CKM.1/DH_PACE** **FCS_CKM.4/PACE**
FCS_COP.1/PACE_MAC	[FDP_ITC.1 or FDP_ITC.2, or FCS_CKM.1], FCS_CKM.4	**FCS_CKM.1/DH_PACE** **FCS_CKM.4/PACE**
FCS_COP.1/PACE_CAM	FDP_ITC.1 or FDP_ITC.2, or FCS_CKM.1], FCS_CKM.4	**FCS_CKM.1/DH_PACE** **FCS_CKM.4/PACE**
FCS_COP.1/PERSO	[FDP_ITC.1 or FDP_ITC.2, or FCS_CKM.1], FCS_CKM.4	**FCS_CKM.1/PERSO** **NA: Perso Keys are not erased in Perso**
FCS_RNG.1/PACE	No dependencies	
FIA_AFL.1/PERSO	FIA_UAU.1	**FIA_UAU.1/PERSO**
FIA_AFL.1/PACE	FIA_UAU.1	**FIA_UAU.1/PACE**
FIA_UID.1/PERSO	No dependencies	
FIA_UAU.1/PERSO	FIA_UID.1	**FIA_UID.1/PERSO**
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	
FDP_RIP.1/PACE	No dependencies	
FTP_ITC.1/PACE	No dependencies	
FMT_SMF.1/PACE	No dependencies	
FMT_SMF.1/PERSO	No dependencies	
FMT_SMR.1/PACE	FIA_UID.1	**FIA_UID.1/PACE**
FMT_SMR.1/PERSO	FIA_UID.1	**FIA_UID.1/PERSO**
FMT_LIM.1/PERSO	FMT_LIM.2	**FMT_LIM.2/PERSO**
FMT_LIM.2/PERSO	FMT_LIM.1	**FMT_LIM.1/PERSO**
FMT_MTD.1/INI_ENA	FMT_SMF.1 FMT_SMR.1	**FMT_SMF.1/PERSO** **FMT_SMR.1/PERSO**
FMT_MTD.1/INI_DIS	FMT_SMF.1 FMT_SMR.1	**FMT_SMF.1/PERSO** **FMT_SMR.1/PERSO**

SFR	Dependencies	Support of the dependencies
FMT_MTD.1/KEY_READ	FMT_SMF.1 FMT_SMR.1	**FMT_SMF.1/PERSO** **FMT_SMR.1/PERSO**
FMT_MTD.1/Initialize_PINPUK	FMT_SMF.1 FMT_SMR.1	**FMT_SMF.1/PERSO** **FMT_SMR.1/PERSO**
FMT_MTD.1/Resume_PINPUK	FMT_SMF.1 FMT_SMR.1	**FMT_SMF.1/PERSO** **FMT_SMR.1/PERSO**
FMT_MTD.1/Change_PIN	FMT_SMF.1 FMT_SMR.1	**FMT_SMF.1/PERSO** **FMT_SMR.1/PERSO**
FMT_MTD.1/Unblock_PIN	FMT_SMF.1 FMT_SMR.1	**FMT_SMF.1/PERSO** **FMT_SMR.1/PERSO**
FPT_EMS.1	No dependencies	
FPT_TST.1	No dependencies	
FPT_FLS.1	No dependencies	
FPT_PHP.3	No dependencies	

Table 25: Security Functional Requirement Dependencies for PACE Module

8.3.6 SFRS DEPENDENCIES for OS Update

SFRs	CC Dependencies	Satisfied Dependencies
FDP_ACC.1/OS-UPDATE	FDP_ACF.1 Security attribute-based access control	FDP_ACF.1/OS-UPDATE
FDP_ACF.1/OS-UPDATE	FDP_ACC.1 Subset access control FMT_MSA.3 Static attribute initialization	FDP_ACC.1/OS-UPDATE FMT_MSA.3/OS-UPDATE
FIA_ATD.1/OS-UPDATE	No Dependencies	No Dependencies
FMT_MSA.3/OSUPDATE	FMT_MSA.1 Management of security attributes FMT_SMR.1 Security roles	See note 1 FMT_SMR.1/OS-UPDATE
FMT_SMR.1/OSUPDATE	FIA_UID.1 Timing of identification	FIA_UID.1/CM
FMT_SMF.1/OS-UPDATE	No Dependencies	No Dependencies
FTP_TRP.1/OS-UPDATE	No Dependencies	No Dependencies
FCS_COP.1/OSUPDATE-DEC	(FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation) FCS_CKM.4 Cryptographic key destruction	FDP_ITC.2/GP-ELF FCS_CKM.4 (from [PPJC])
FPT_FLS.1/OS-UPDATE	No Dependencies	No Dependencies
FCS_COP.1/OSUPDATE-VER	(FDP_ITC.1 Import of user data without security attributes, or FDP_ITC.2 Import of user data with security attributes, or FCS_CKM.1 Cryptographic key generation) FCS_CKM.4 Cryptographic key destruction	FDP_ITC.2/Installer FCS_CKM.4 (from [PPJC])

Table 26: SFR Dependencies for OS Update

Note 1: The dependency **FMT_MSA.1** of **FMT_MSA.3/OS-UPDATE** is unsupported. No history information has to be kept by the TOE.

8.3.7 Compatibility between SFR of TOE and SFR of [IFX-IC]

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

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

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

The first column lists the [IFX-IC] and the next columns indicate their classification according to the paragraph above. The SFR's on the cells of the classification belong the MultiApp v4.2 TOE described in this document. If there is no SFR on each cell is because not all CC class families have a corresponding match on both sides, but all SFRs from the [IFX-IC] have been classified. Moreover, no contradictions have been found between the Platform-SFRs set and the SFRs related to the composite product.

IFX_CCI_000043h SFR's	IP_SFR	RP_SFR-SERV (*)	RP_SFR-MECH
Security functional requirements of the TOE defined in [PP0084]			
FRU_FLT.2			X FDP_SDI.2/DATA FPT_PHP.3/SCP
FPT_FLS.1			X FPT_FLS.1/JCS FPT_FLS.1/Installer FPT_FLS.1/ADEL FPT_FLS.1/ODEL FPT_FLS.1/SpecificAPI
FMT_LIM.1		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FMT_LIM.2		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FAU_SAS.1	X		
FDP_SDC.1			X *see table 28
FDP_SDI.2			X FDP_SDI.2/DATA
FPT_PHP.3			X FPT_PHP.3/SCP
FDP_ITT.1			X *see table 28

IFX_CCI_000043h SFR's	IP_SFR	RP_SFR-SERV (*)	RP_SFR-MECH
FPT_ITT.1		X FPT_ITT.1/SpecificAPI	
FDP_IFC.1			X FDP_IFC.1/JCVM
FCS_RNG.1/TRNG			x FCS_RNG.1
FCS_RNG.1/DRNG			x FCS_RNG.1
FCS_RNG.1/DRNG4			x FCS_RNG.1
FCS_RNG.1/RCL/TRNG	X		
FCS_RNG.1/RCL/DRNG3	X		
FCS_RNG.1/RCL/DRNG4	X		
FCS_RNG.1/HPRG			x FCS_RNG.1
FCS_COP.1/SCP/TDES			X FCS_COP.1
FCS_CKM.4/SCP		X FCS_CKM.4	
FCS_COP.1/SCP/AES			X FCS_COP.1
FCS_COP.1/SCL/TDES	X		
FCS_CKM.4/SCL	X		
FCS_COP.1/SCL/AES	X		
FMT_LIM.1/Loader		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FMT_LIM.2/Loader		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FTP_ITC.1		X FTP_ITC.1/CM FTP_ITC.1/PACE	
FDP_UCT.1		X FDP_ACC.1.1/CMGR	
FDP_UIT.1			X FDP_UIT.1/CM
FDP_ACC.1/Loader		X FDP_ACC.1.1/CMGR	

IFX_CCI_000043h SFR's	IP_SFR	RP_SFR-SERV (*)	RP_SFR-MECH
FDP_ACF.1/Loader		X FDP_ACF.1.1/CMGR	
FIA_API.1			X FIA_UID.2/AID FIA_UID.1/CM
Additional security functional requirements of the TOE			
FPT_TST.2			X FPT_TST.1/SCP
FDP_ACC.1		X FDP_ACC.1.1/CMGR	
FDP_ACF.1		X FDP_ACF.1.1/CMGR	
FMT_MSA.1		X FMT_MSA.1/ADEL FMT_MSA.1/JCRE FMT_MSA.1/JCVM FMT_MSA.1/CM FMT_MSA.1/CMGR	
FMT_MSA.3		X FMT_MSA.3/FIREWALL FMT_MSA.3/JCVM FMT_MSA.3/ADEL FMT_MSA.3/CM FMT_MSA.3/CMGR	
FMT_SMF.1		X FMT_SMF.1/CORE_LC FMT_SMF.1/ADEL FMT_SMF.1/CM	
FMT_SMR.1			
FCS_COP.1/SCL/TDES-MAC	X		
FCS_COP.1/SCL/AES-MAC	X		
FCS_COP.1/RSA/<iteration>			X FCS_COP.1
FCS_CKM.1/RSA/<iteration>		X FCS_CKM.1	
FCS_CKM.4/RSA	X		
FCS_COP.1/ECC/<iteration>			X FCS_COP.1
FCS_CKM.1/ECC		X FCS_CKM.1	
FCS_CKM.4/ECC	X		
FCS_COP.1/HCL	X		

IFX_CCI_000043h SFR's	IP_SFR	RP_SFR-SERV (*)	RP_SFR-MECH
FMT_MTD.1/Loader		X FMT_MTD.1/INI_ENA Management of TSF data, FMT_MTD.1/JCRE	
FMT_SMR.1/Loader		X FMT_SMR.1/PERSO, FMT_SMR.1.1/CM, FMT_SMR.1.1/Installer, FMT_SMR.1/JCRE	
FMT_SMF.1/Loader		X FMT_SMF.1/CM	
FIA_UID.2/Loader		X FIA_UID.2/AID	

Table 27 Compatibility between SFR of TOE and SFR of [IFX-IC]

8.3.8 Compatibility between SFR of PACE MODULE and [IFX-IC]

The format of the format of the following table follows the same principle as the one on previous section.

IFX_CCI_000043h SFR's	IP_SFR	RP_SFR-SERV (*)	RP_SFR-MECH
Security functional requirements of the TOE defined in [PP0084]			
FRU_FLT.2			X FPT_PHP.3
FPT_FLS.1			X FPT_FLS.1
FMT_LIM.1		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FMT_LIM.2		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FAU_SAS.1	X		
FDP_SDC.1			X FPT_EMS.1
FDP_SDI.2			X *see table 27
FPT_PHP.3			X FPT_PHP.3
FDP_ITT.1			X FPT_EMS.1
FPT_ITT.1		X *see table 26	
FDP_IFC.1			X *see table 27
FCS_RNG.1/TRNG			x FCS_RNG.1/PACE
FCS_RNG.1/DRNG			x FCS_RNG.1/PACE
FCS_RNG.1/DRNG4			x FCS_RNG.1/PACE
FCS_RNG.1/HPRG			x FCS_RNG.1/PACE
FCS_RNG.1/RCL/TRNG	X		
FCS_RNG.1/RCL/DRNG3	X		
FCS_RNG.1/RCL/DRNG4	X		

IFX_CCI_000043h SFR's	IP_SFR	RP_SFR-SERV (*)	RP_SFR-MECH
FCS_COP.1/SCP/TDES			X FCS_COP.1/PACE_ENC FCS_COP.1/PERSO
FCS_CKM.4/SCP		X FCS_CKM.4 /PACE	
FCS_COP.1/SCP/AES			X FCS_COP.1/PACE_ENC FCS_COP.1/PERSO
FCS_COP.1/SCL/TDES	X		
FCS_CKM.4/SCL	X		
FCS_COP.1/SCL/AES	X		
FCS_CKM.4/SCL/AES	X		
FMT_LIM.1/Loader		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FMT_LIM.2/Loader		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FTP_ITC.1		X *see table 26	
FDP_UCT.1		X *see table 26	
FDP_UIT.1			X *see table 27
FDP_ACC.1/Loader		X *see table 26	
FDP_ACF.1/Loader		X *see table 26	
FIA_API.1			X FIA_UID.1/PERSO FIA_UAU.1/PERSO FIA_UID.1/PACE FIA_UAU.1/PACE FIA_UAU.4/PACE FIA_UAU.5/PACE FIA_UAU.6/PACE
Additional security functional requirements of the TOE			
FPT_TST.2			X FPT_TST.1
FDP_ACC.1		X *see table 26	

IFX_CCI_000043h SFR's	IP_SFR	RP_SFR-SERV (*)	RP_SFR-MECH
FDP_ACF.1		X *see table 26	
FMT_MSA.1		X FPT_TST.1	
FMT_MSA.3		X FPT_TST.1	
FMT_SMF.1		X FMT_SMF.1/PACE FMT_SMF.1/PERSO	
FMT_SMR.1		FMT_SMR.1/PACE FMT_SMR.1/PERSO	
FCS_COP.1/SCP/TDES-MAC			X FCS_COP.1/PACE_MAC FCS_COP.1/PERSO
FCS_COP.1/SCP/AES-MAC			X FCS_COP.1/PACE_MAC FCS_COP.1/PERSO
FCS_COP.1/RSA/<iteration>	X		
FCS_CKM.1/RSA/<iteration>	X		
FCS_CKM.4/RSA	X		
FCS_COP.1/ECC/<iteration>			X FCS_COP.1/PACE_CAM
FCS_CKM.1/ECC		X FCS_CKM.1/DH_PACE	
FCS_CKM.4/ECC	X		
FCS_COP.1/HCL	X		
FMT_MTD.1/Loader	X		
FMT_SMR.1/Loader	X		
FMT_SMF.1/Loader	X		
FIA_UID.2/Loader	X		

Table 28 Compatibility between SFR of PACE MODULE and [IFX-IC]

(*) RP_SFR-SERV group definition:

8.3.9 SAR DEPENDENCIES

Requirements	CC dependencies	Satisfied dependencies
ADV_ARC.1	ADV_FSP.1; ADV_TDS.1	ADV_FSP.5; ADV_TDS.5
ADV_FSP.5	ADV_TDS.1; ADV_IMP.1	ADV_TDS.5; ADV_IMP.2
ADV_IMP.2	ADV_TDS.3; ALC_CMC.5; ALC_TAT.1	ADV_TDS.5; ALC_CMC.5; ALC_TAT.3
ADV_INT.3	ADV_IMP.1; ADV_TDS.3; ALC_TAT.1	ADV_IMP.2; ADV_TDS.5; ALC_TAT.3
ADV_SPM.1	ADV_FSP.4	ADV_FSP.5
ADV_TDS.5	ADV_FSP.5	ADV_FSP.5
AGD_OPE.1	ADV_FSP.1	ADV_FSP.5
AGD_PRE.1	None	
ALC_CMC.5	ALC_CMS.1; ALC_DVS.2; ALC_LCD.1	ALC_CMS.5; ALC_DVS.2; ALC_LCD.1
ALC_CMS.5	None	
ALC_DEL.1	None	
ALC_DVS.2	None	
ALC_LCD.1	None	
ALC_FLR.1	None	
ALC_TAT.3	ADV_IMP.1	ADV_IMP.2
ATE_COV.3	ADV_FSP.2; ATE_FUN.1	ADV_FSP.5; ATE_FUN.2
ATE_DPT.3	ADV_ARC.1; ADV_TDS.4; 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; AGD_OPE.1; AGD_PRE.1; ATE_COV.1; ATE_FUN.1	ADV_FSP.5; AGD_OPE.1; AGD_PRE.1; ATE_COV.2; ATE_FUN.2
AVA_VAN.5	ADV_ARC.1; ADV_FSP.4; ADV_TDS.3; ADV_IMP.1; AGD_OPE.1; AGD_PRE.1; ATE_DPT.1	ADV_ARC.1; ADV_FSP.5; ADV_TDS.5; ADV_IMP.2; AGD_OPE.1; AGD_PRE.1; ATE_DPT.3

Table 29: SAR dependencies for EAL6

8.3.10 RATIONALE FOR THE SECURITY ASSURANCE REQUIREMENTS

8.3.10.1 EAL6: semiformally verified design and tested

EAL6 is required for this type of TOE and product since it is intended to defend against sophisticated attacks. This evaluation assurance level allows a developer to gain high assurance from application of security engineering techniques to a rigorous development environment in order to produce a premium TOE for protecting high value assets against significant risks.
The evaluators should have access to the a formal model of select TOE security policies and a semiformal presentation of the functional specification and TOE low level design and source code.

8.3.10.2 ALC_FLR.1 Basic flaw remediation

This augmentation claim in this Security Target will cover the policies and procedures applied to track and correct flaws and support surveillance of this TOE.

9 TOE SUMMARY SPECIFICATION

9.1 TOE SECURITY FUNCTIONS

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

9.1.1 SF provided by MultiApp V5.2 platform

9.1.1.1 SF.FW: Firewall

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

An operation OP.PUT (S1, S.MEMBER, I) is allowed if and only if the active context is "JCRE"; other OP.PUT operations are allowed regardless of the active context's value.	FDP_IFC.1/JCVM FDP_IFF.1/JCVM
Upon allocation of a resource to class instances and arrays, any previous information content of the resource is made unavailable	FDP_RIP.1/OBJECTS
Only the S.JCRE can modify the security attributes the active context, the selected applet context security attributes.	FMT_MSA.1/JCRE
Only the S.JCVM can modify the security attributes the active context, the currently active Context and the Active Applets security attributes.	FMT_MSA.1/JCVM
provide restrictive default values for security attributes that are used to enforce the SFP.	FMT_MSA.3/JCVM
only secure values are accepted for all the security attributes of subjects and objects defined in the FIREWALL access control SFP and the JCVM information flow control SFP.	FMT_MSA.2/FIREWALL_ JCVM
provide restrictive default values for security attributes that are used to enforce the SFP.	FMT_MSA.3/FIREWALL
The TSF maintains the roles: the Java Card RE, the Java Card VM. The TSF is able to associate users with roles.	FMT_SMR.1/JCRE
The TSF is capable of performing the following management functions: • Modify the active context and the SELECTed applet Context. • Modify the list of registered applets' AID	FMT_SMF.1/CORE_LC
([JCRE3]§6.2.8) An S.CAP_FILE may freely perform any of OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW or OP.TYPE_ACCESS upon any O.JAVAOBJECT whose Sharing attribute has value "JCRE entry point" or "global array".	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
([JCRE3]§6.2.8) An S.CAP_FILE may freely perform any of OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE or OP.THROW upon any O.JAVAOBJECT whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if O.JAVAOBJECT's Context attribute has the same value as the active context.	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
([JCRE3]§6.2.8.10) An S.CAP_FILE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT whose Sharing attribute has value "SIO" only if O.JAVAOBJECT is being cast into (checkcast) or is being verified as being an instance of (instanceof) an interface that extends the Shareable interface.	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
• ([JCRE3], §6.2.8.6,) An S.CAP_FILE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if one of the following applies: (c) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable",	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL

(d) The value of the attribute Selection Status of the package whose AID is "Package AID' is "Non-multiselectable", and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute ActiveApplets, and in either of the cases above the invoked interface method extends the Shareable interface	
An S.CAP_FILE may perform an OP.CREATE only if the value of the Sharing parameter(*) is "Standard".	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
The subject S.JCRE can freely perform OP.JAVA(...) and OP.CREATE, with the following two exceptions: 1. Any subject with OP.JAVA upon an O.JAVAOBJECT whose LifeTime attribute has value "CLEAR_ON_DESELECT" if O.JAVAOBJECT's Context attribute is not the same as the SELECTed applet Context. 2. Any subject with OP.CREATE and a "CLEAR_ON_DESELECT" LifeTime parameter if the active context is not the same as the SELECTed applet Context.	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
The TSF allows the rollback of the operations OP.JAVA and OP.CREATE on the O.JAVAOBJECTs.	FDP_ROL.1/FIREWALL
The TSF allows operations to be rolled back within the scope of a select(), deselect(), process() or install() call, notwithstanding the restrictions given in [JCRE3], §7.7, within the bounds of the Commit Capacity ([JCRE3], §7.8), and those described in [JCAPI3].	FDP_ROL.1/FIREWALL
Only updates to persistent objects participate in the transaction. Updates to transient objects and global arrays are never undone, regardless of whether or not they were “inside a transaction.” [JCRE3], §7.7	FDP_ROL.1/FIREWALL
A TransactionException is thrown if the commit capacity is exceeded during a transaction. [JCRE3], §7.8	FDP_ROL.1/FIREWALL
Transaction & PIN: When comparing a PIN, even if a transaction is in progress, update of internal state - the try counter, the validated flag, and the blocking state, do not participate in the transaction. [JCAPI3]	FDP_ROL.1/FIREWALL

9.1.1.2 SF.API: Application Programming Interface

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

- TDES algorithm support 112-bit key and 168-bit key
- RSA algorithm supports up to 4096 bit keys (Std method or CRT method).
- AES algorithm with 128, 192 and 256 bit keys.
- Random generator uses the certified Hardware Random Generator that fulfils the requirements of AIS31 (see [ST_IC]).
- SHA-1, SHA224, SHA-256, SHA-384, SHA-512 and SHAKE256 algorithms
- Diffie-Hellman based on exponentiation and on EC algorithm.
- PACE based on DH algorithm (integrated mapping and generic mapping)
- PACE based on ECDH algorithm (integrated mapping and generic mapping)
- ML-DSA-65 (level3)

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

- Key generation: The TOE provides the following:
  - RSA key generation manages 1024 to 2048-bits long keys. The RSA key generation is SW and does not use the IC cryptographic library.
  - The TDES key generation (for session keys) uses the random generator.
  - AES key generation
  - DH key generation
  - ECDH key generation
  - ML-DSA key generation

- Key destruction: the TOE provides a specified cryptographic key destruction method that makes Key unavailable.

This security function ensures the confidentiality of keys during manipulation and ensures the de-allocation of memory after use.

This security function is supported by the IC security function SF.CS (Cryptographic support) for Random Number Generator (see [ST_IC]).

RSA standard Key generation Algorithm - 1024,1536,2048	FCS_CKM.1
RSA CRT Key generation Algorithm - 1024,1536,2048, 3072, 4096	FCS_CKM.1
AES Key generation Algorithm - 128, 192, 256	FCS_CKM.1
ECC Key generation Algorithm - 160, 192, 224, 256, 320, 384, 512, 521	FCS_CKM.1
EC Diffie-Hellman Key agreement Algorithm 160, 192, 224, 256, 320, 384, 512, 521	FCS_CKM.1
DH Key agreement Algorithm 1024, 1280,1536, 2048	FCS_CKM.1
ML-DSA-65 Key Generation level 3	FCS_CKM.1
Key distribution with JC API setkey()	FCS_CKM.2
Key access with JC API getkey()	FCS_CKM.3
Key deletion with JC API clearkey()	FCS_CKM.4
RSA standard Signature & Verification – RSA SHA PKCS#1, RSA SHA PKCS#1 PSS – 1024,1152,1280,1536,2048	FCS_COP.1
RSA CRT Signature & Verification – RSA SHA PKCS#1, RSA SHA PKCS#1 PSS 1024,1152,1280,1536,2048, 3072, 4096	FCS_COP.1
RSA standard Encryption & Decryption – 1536, 1792, 2048	FCS_COP.1
RSA CRT Encryption & Decryption – 1024,1152,1280,1536,2048, 3072, 4096	FCS_COP.1
TDES Encryption & Decryption – DES NOPAD, DES PKCS#5, DES 9797 M1 M2 – 112, 168	FCS_COP.1
TDES Signature & Verification – DES MAC ISO9797-1 M1 M2, DES MAC NOPAD, DES MAC PKCS#5- 112, 168	FCS_COP.1
AES Encryption & Decryption – AES 128 NOPAD – 128, 192, 256	FCS_COP.1
AES Signature & Verification – AES MAC 128 NOPAD – 128, 192, 256	FCS_COP.1
ECDSA Signature & Verification – ECDSA SHA – 160, 192, 224, 256, 320, 384, 512, 521	FCS_COP.1
SHA-1, SHA2 (224, 256, 384, 512) , SHA3 (224, 256, 384, 512) Message digest SHAKE256	FCS_COP.1
ECC for PACE Integrited Mapping & Generic Mapping 160, 192, 224, 256, 320, 384, 512, 521	FCS_COP.1
DH for PACE Integrited Mapping & Generic Mapping 1024, 2048	FCS_COP.1
ECC for Pseudonym signature 160, 192, 224, 256, 320, 384, 512, 521	FCS_COP.1
ML-DSA-65 Key generation & Signature/Verification level 3	FCS_COP.1

9.1.1.3 SF.CSM: Card Security Management

Upon allocation of a resource to the APDU buffer, any previous information content of the resource is made unavailable.	FDP_RIP.1/APDU
Upon deallocation of a resource from the bArray object, any previous information content of the resource is made unavailable.	FDP_RIP.1/bArray
Upon deallocation of a resource from any reference to an object instance created during an aborted transaction, any previous information content of the resource is made unavailable.	FDP_RIP.1/ABORT
Upon deallocation of a resource from the cryptographic buffer (D.CRYPTO), any previous information content of the resource is made unavailable.	FDP_RIP.1/KEYS
Upon deallocation of a resource from the applet as a result of returning from the process method to the following objects: a user Global Array, any previous information content of the resource is made unavailable.	FDP_RIP.1/GlobalArray
Upon deallocation of a resource from the transient object, any previous information content of the resource is made unavailable.	FDP_RIP.1/TRANSIENT
The TSF takes the following actions: • throw an exception,	FAU_ARP.1

• or lock the card session • or reinitialize the Java Card System and its data upon detection of a potential security violation.	
The TOE detects the following potential security violation: • CAP file inconsistency • Applet life cycle inconsistency • Card Manager life cycle inconsistency • Card tearing (unexpected removal of the Card out of the CAD) and power failure • Abortion of a transaction in an unexpected context (see abortTransaction(), [JCAPI3] and ([JCRE3], §7.6.2) • Violation of the Firewall or JCVM SFPs • Unavailability of resources • Array overflow • Random trap detection	FAU_ARP.1
The TSF is able to monitor user data stored in containers controlled by the TSF for **integrity errors** on all the following objects: Cryptographic keys, PINs, applets, and softmasks when they are stored in EEPROM. Upon detection of a data integrity error, the TSF: • Prevents the use of modified data • Raises an exception	FDP_SDI.2/DATA
In order to consistently interpret **the CAP files, the bytecode and its data argument,** when shared between the TSF and another trusted IT product, the TSF is using: • The rules defined in [JCVM3] specification; • The API tokens defined in the export files of reference implementation • **The rules defined in ISO 7816-6** • **The rules defined in [GP23] specification**	FPT_TDC.1
The TSF preserves a secure state when the following types of failures occur: **those associated to the potential security violations described in FAU_ARP.1.** The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE3], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE3] §4.1.2). Behavior of the TOE on power loss and reset is described in [JCRE3], §3.6, and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE3], §3.6.2	FPT_FLS.1/JCS
No one can observe the operation **cryptographic operations / comparisons operations** on **Key values / PIN values** by **S.JCRE, S.Applet.**	FPR_UNO.1

9.1.1.4 SF.AID: AID Management

Only **the JCRE** can **modify** the **list of registered applets' AIDs**.	FMT_MTD.1/JCRE
Only secure values are accepted for **the AIDs of registered applets**.	FMT_MTD.3/JCRE
The TSF maintains the following list of security attributes belonging to individual users: • **package AID** • **Applet's version number** • **registered applet's AID** • **applet selection status ([JCVM3], §6.5)**	FIA_ATD.1/AID
The TSF requires each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.	FIA_UID.2/AID
Initial applet selection is performed as described in [JCRE3]§4 **Applet selection is performed after a successful SELECT FILE command as described in [JCRE3]§4**.	FIA_USB.1/AID

9.1.1.5 SF.INST: Installer

the protocol used provides for the unambiguous association between the security attributes and the user data received: The format of the CAP file is precisely defined in Sun's specification ([JCVM3]); it contains the user data (like applet's code and data) and the security attribute altogether.	FDP_ITC.2/Installer
Each package contains a package Version attribute, which is a pair of major and minor version numbers ([JCVM3], §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 ([JCVM3], §4.5.2): the dependent packages Versions and AIDs attributes allow the retrieval of these identifications.. Implementation-dependent checks may occur on a case-by-case basis to indicate that package files are binary compatibles. However, package files do have "package Version Numbers" ([JCVM3]) used to indicate binary compatibility or incompatibility between successive implementations of a package, which obviously directly concern this requirement.	FDP_ITC.2/Installer
A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs. The loading is allowed only if, for each dependent package, its AID attribute is equal to a resident package AID attribute, the major (minor) Version attribute associated to the former is equal (less than or equal) to the major (minor) Version attribute associated to the latter ([JCVM3],§4.5.2).	FDP_ITC.2/Installer
The TSF maintains the roles**: the installer**	FMT_SMR.1/Installer
The TSF preserves a secure state when the following types of failures occur: **the installer fails to load/install a package/applet as described in [JCRE3] §11.1.4**	FPT_FLS.1/Installer
After **Failure during applet loading, installation and deletion; sensitive data loading**, the TSF ensures the return of the TOE to a secure state using automated procedures. The TSF provides the capability to determine the objects that were or were not capable of being recovered.	FPT_RCV.3/Installer

9.1.1.6 SF.ADEL: Applet Deletion

Only **the Java Card RE (S.JCRE)** can **modify** the security attributes: **ActiveApplets**. The modification of the ActiveApplets security attribute should be performed in accordance with the rules given in [JCRE3], §4.	FMT_MSA.1/ADEL
Provide restrictive default values for security attributes that are used to enforce the SFP.	FMT_MSA.3/ADEL
The TSF maintains the roles: **the applet deletion manager**.	FMT_SMR.1/ADEL
The TSF is able to **Modify the ActiveApplets security attribute**.	FMT_SMF.1/ADEL
([JCRE3], §11.3.4.1, Applet Instance Deletion). The S.ADEL may perform OP.DELETE_APPLET upon an O.APPLET only if, (1) S.ADEL is currently selected, (2) O.APPLET is deselected and (3) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance distinct from O.APPLET, or O.JAVAOBJECT is reachable from a package P, or ([JCRE3], §8.5) O.JAVAOBJECT is remote reachable.	FDP_ACC.2/ADEL FDP_ACF.1/ADEL
([JCRE3], §11.3.4.1, Multiple Applet Instance Deletion). The S.ADEL may perform OP.DELETE_APPLET upon several O.APPLET only if, (1) S.ADEL is currently selected, (2) every O.APPLET being deleted is deselected and (3) there is no O.JAVAOBJECT owned by any of the O.APPLET being deleted such that either O.JAVAOBJECT is reachable from an applet instance distinct from any of those O.APPLET, or O.JAVAOBJECT is	FDP_ACC.2/ADEL FDP_ACF.1/ADEL

reachable from a package P, or ([JCRE3], §8.5) O.JAVAOBJECT is remote reachable.	
([JCRE3], §11.3.4.2, Applet/Library Package Deletion). The S.ADEL may perform OP.DELETE_CAP_FILE upon an O.CODE_CAP_FILE only if, (1) S.ADEL is currently selected, (2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_CAP_FILE that is an instance of a class that belongs to O.CODE_CAP_FILE exists on the card and (3) there is no package loaded on the card that depends on O.CODE_CAP_FILE.	FDP_ACC.2/ADEL FDP_ACF.1/ADEL
([JCRE3], §11.3.4.3, Applet Package and Contained Instances Deletion). The S.ADEL may perform OP.DELETE_CAP_FILE_APPLET upon an O.CODE_CAP_FILE only if, (1) S.ADEL is currently selected, (2) no reachable O.JAVAOBJECT, from a package distinct from O.CODE_CAP_FILE, which is an instance of a class that belongs to O.CODE_CAP_FILE exists on the card, (3) there is no package loaded on the card that depends on O.CODE_CAP_FILE and (4) for every O.APPLET of those being deleted it holds that: (i) O.APPLET is deselected and (ii) there is no O.JAVAOBJECT owned by O.APPLET such that either O.JAVAOBJECT is reachable from an applet instance not being deleted, or O.JAVAOBJECT is reachable from a package not being deleted, or ([JCRE3],§8.5) O.JAVAOBJECT is remote reachable.	FDP_ACC.2/ADEL FDP_ACF.1/ADEL
However, the S.ADEL may be granted privileges ([JCRE3], §11.3.5) to bypass the preceding policies. For instance, the logical deletion of an applet renders it un-selectable; this has implications on the management of the associated TSF data (see application note of FMT_MTD.1.1/JCRE).	FDP_ACF.1/ADEL
Only the S.ADEL can delete O.CODE_CAP_FILE or O.APPLET from the card.	FDP_ACF.1/ADEL
Upon deallocation of a resource from the **applet instances and/or packages when one of the deletion operations in FDP_ACC.2.1/ADEL is performed on them**, any previous information content of the resource is made unavailable.	FDP_RIP.1/ADEL
Requirements on de-allocation during applet/package deletion are described in [JCRE3], §11.3.4.1, §11.3.4.2 and §11.3.4.3.	FDP_RIP.1/ADEL
The TSF preserves a secure state when the following types of failures occur: **the applet deletion manager fails to delete a package/applet as described in [JCRE3], §11.3.4**.	FPT_FLS.1/ADEL

9.1.1.7 SF.ODEL: Object Deletion

Upon deallocation of the resource from the objects owned by the context of an applet instance which triggered the execution of the method javacard.framework.JCSystem.requestObjectDeletion(), any previous information content of the resource is made unavailable.	FDP_RIP.1/ODEL
The TSF preserves a secure state when the following types of failures occur: the object deletion functions fail to delete all the unreferenced objects owned by the applet that requested the execution of the method.	FPT_FLS.1/ODEL

9.1.1.8 SF.CAR: Secure Carrier

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

9.1.1.9 SF.SCP: Smart Card Platform

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

9.1.1.10 SF.CMG: Card Manager

The Card Manager loads and extradites applets. It also loads GP key.	FDP_ACC.1/CMGR FDP_ACF.1/CMGR
No one can modify the security attribute code category	FMT_MSA.1/CMGR
Only restrictive default values can be used for the code category	FMT_MSA.3/CMGR

9.1.1.11 SF.APIs: Specific API

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

9.1.1.12 SF.RND: RNG

Provide a random value	FCS_RNG.1

9.1.1.13 *SF.OSAGILITY: OS Agility Management*

Provides the role management as defined in	FMT_SMR.1/OS-UPDATE
Provides Patch management functions linked to the states of the TOE as defined in	FMT_SMF.1/OS-UPDATE
The TSF maintains the following list of security attributes belonging to individual users: • **additional code ID for each activated additional code**	FIA_ATD.1/OS-UPDATE
The OS Update module load, install and activate the additional code	FDP_ACC.1/OS-UPDATE FDP_ACF.1/OS-UPDATE
The default values for security attributes are defined by the OS Update Access Control Policy	FMT_MSA.3/OS-UPDATE
Provides a communication path between itself and remote	FTP_TRP.1/OS-UPDATE
It provides the secure transfer of data through SM as defined in	FCS_COP.1/OS-UPDATE-VER FCS_COP.1/OS-UPDATE-DEC
Provides physical protection of the TOE and preservation of TOE secure state as defined in	FPT_FLS.1/OS-UPDATE

9.1.2 SF provided by MultiApp V5.2 PACE Module

SF	Description
SF.REL	Protection of data
SF.AC	Access control
SF.SYM_AUTH	Symmetric authentication
SF.SM	Secure messaging
SF.PERSO	Provides service for Personalization of data in used in PACE

Table 30: Security Functions provided by the MultiApp V5.2 with PACE

The SF.REL function provides the protection of data on the TOE as detailed in next table.

Provides physical protection of the TOE and preservation of TOE secure state as defined in	**FPT_PHP.3**, **FPT_FLS.1**
Addresses the inherent leakage to TOE cryptographic operation	**FPT_EMS.1**
Provides the TOE test mechanisms as defined in	**FPT_TST.1**
Provides protection against misuse of TOE test features as defined in	**FMT_LIM.1/PERSO** and **FMT_LIM.2/PERSO**

The SF.AC function provides the access control of the TOE as listed in next table.

Provides TOE access control to specific data as defined in	**FMT_MTD.1/INI_ENA**, **FMT_MTD.1/INI_DIS**, **FMT_MTD.1/Initialize_PINPUK, FMT_MTD.1/Resume_PINPUK, FMT_MTD.1/Change_PIN, FMT_MTD.1/Unblock_PIN**
Provides no access to specific data as defined in	**FMT_MTD.1/KEY_READ**
Provides the role management as defined in	**FMT_SMR.1/PACE FMT_SMR.1/PERSO**
Provides management functions linked to the states of the TOE as defined in	**FMT_SMF.1/PERSO FMT_SMF.1/PACE**

The SF.SYM_AUTH function provides the symmetric authentication functions to the TOE as listed in next table.

It encompasses the PACE identification and authentication as defined in	**FIA_UID.1/PACE FIA_UAU.1/PACE FIA_UAU.4/PACE FIA_UAU.5/PACE FIA_UAU.6/PACE**
It manages error in SM establishment as defined in	**FIA_AFL.1/PACE**
The role authentication as requested by	**FMT_SMR.1/PACE**

The SF.SM function provides the secure messaging of the TOE as listed in next table.

It provides the establishment of SM as defined in	**FCS_CKM.1/DH_PACE**, **FTP_ITC.1/PACE**, **FCS_RNG.1/PACE**
It provides the secure transfer of data through SM as defined in	**FCS_COP.1/PACE_ENC FCS_COP.1/PACE_MAC FCS_COP.1/PACE_CAM**
It performs the erasure of session keys and sensitive data as defined in	**FCS_CKM.4/PACE** and **FDP_RIP.1/PACE.**

The SF.PERSO function provides the service to personalize the TOE as listed in next table.

It provides the nonce and session key for SM for personalization operation as defined in	**FCS_RNG.1/PACE, FCS_CKM.1/PERSO**
It provides the establishment of SM and manage error as defined in	**FIA_AFL.1/PERSO**
It provides the identification and authentication in personalisation phase as defined in	**FIA_UID.1/PERSO FIA_UAU.1/PERSO FMT_SMR.1/PERSO**
It provides secure import of sensitive data using crypto mechanisms	**FMT_SMF.1/PERSO FCS_COP.1/PERSO**
It performs the erasure of session keys and sensitive data as defined in	**FCS_CKM.4/PACE** and **FDP_RIP.1/PACE**

9.1.3 TSFs provided by the IFX_CCI_000043h

The evaluation is a composite evaluation and uses the results of the CC evaluation provided by [CR-IC]. The IC and its primary embedded software have been evaluated at level EAL 6+. These SF are the same for the IC considered in this ST;

SF	Description
SF_DPM	Device Phase Management
SF_PS	Protection against Snooping
SF_PMA	Protection against Modification Attacks
SF_PLA	Protection against Logical Attacks
SF_CS	Cryptographic Support

Table 31: Security Functions provided by the Thales Design Services IFX_CCI_000043h chips

These SF are described in [IFX-IC].

10 RATIONALES

10.1 TOE SUMMARY SPECIFICATION RATIONALE

10.1.1 TOE security functions rationale

	SF.FW	SF.API	SF.CSM	SF.AID	SF.INST	SF.ADEL	SF.ODEL	SF.CAR	SF.SCP	SF.CMG	SF.APIs.	SF.RND	SF.OSAGILITY
FDP_ACC.2/FIREWALL	X												
FDP_ACF.1/FIREWALL	X												
FDP_IFC.1/JCVM	X												
FDP_IFF.1/JCVM	X												
FDP_RIP.1/OBJECTS	X												
FMT_MSA.1/JCRE	X												
FMT_MSA.1/JCVM	X												
FMT_MSA.2/FIREWALL_JCVM	X												
FMT_MSA.3/FIREWALL	X												
FMT_MSA.3/JCVM	X												
FMT_SMR.1/JCRE	X												
FMT_SMF.1/CORE_LC	X												
FCS_CKM.1		X											
FCS_CKM.2		X											
FCS_CKM.3		X											
FCS_CKM.4		X											
FCS_COP.1		X											
FDP_RIP.1/ABORT			X										
FDP_RIP.1/APDU			X										
FDP_RIP.1/GlobalArray			X										
FDP_RIP.1/bArray			X										
FDP_RIP.1/KEYS			X										
FDP_RIP.1/TRANSIENT			X										
FDP_ROL.1/FIREWALL	X												
FAU_ARP.1			X										
FDP_SDI.2/DATA			X										
FPR_UNO.1			X										
FPT_FLS.1/JCS			X										
FPT_TDC.1			X										
FIA_ATD.1/AID				X									
FIA_UID.2/AID				X									
FIA_USB.1/AID				X									
FMT_MTD.1/JCRE				X									
FMT_MTD.3/JCRE				X									
FDP_ITC.2/Installer					X								
FMT_SMR.1/Installer					X								
FPT_FLS.1/Installer					X								
FPT_RCV.3/Installer					X								
FDP_ACC.2/ADEL						X							
FDP_ACF.1/ADEL						X							
FDP_RIP.1/ADEL						X							
FMT_MSA.1/ADEL						X							
FMT_MSA.3/ADEL						X							
FMT_SMF.1/ADEL						X							
FMT_SMR.1/ADEL						X							

	SF.FW	SF.API	SF.CSM	SF.AID	SF.INST	SF.ADEL	SF.ODEL	SF.CAR	SF.SCP	SF.CMG	SF.APIs.	SF.RND	SF.OSAGILITY
FPT_FLS.1/ADEL						X							
FDP_RIP.1/ODEL							X						
FPT_FLS.1/ODEL							X						
FCO_NRO.2/CM								X					
FDP_IFC.2/CM								X					
FDP_IFF.1/CM								X					
FDP_UIT.1/CM								X					
FIA_UID.1/CM								X					
FMT_MSA.1/CM								X					
FMT_MSA.3/CM								X					
FMT_SMF.1/CM								X					
FMT_SMR.1/CM								X					
FTP_ITC.1/CM								X					
FPT_TST.1/SCP									X				
FPT_PHP.3/SCP									X				
FPT_RCV.4/SCP									X				
FDP_ACC.1/CMGR										X			
FDP_ACF.1/CMGR										X			
FMT_MSA.1/CMGR										X			
FMT_MSA.3/CMGR										X			
FPT_FLS.1/SpecificAPI											X		
FPT_ITT.1/SpecificAPI											X		
FPR_UNO.1/SpecificAPI.											X		
FCS_RNG.1												X	
FMT_SMR.1/OS-UPDATE													X
FMT_SMF.1/OS-UPDATE													X
FIA_ATD.1/OS-UPDATE													X
FDP_ACC.1/OS-UPDATE													X
FDP_ACF.1/OS-UPDATE													X
FMT_MSA.3/OS-UPDATE													X
FTP_TRP.1/OS-UPDATE													X
FCS_COP.1/OS-UPDATE-VER													X
FCS_COP.1/OS-UPDATE-DEC													X
FPT_FLS.1/OS-UPDATE													X

Table 32: Rationale table of functional requirements and security functions

10.1.2 TOE security functions rationale for PACE MODULE

	SF.REL	SF.AC	SF.SYM_AUT	SF.SM	SF.PERSO	SF of IFX_CCI_000043h	SF_DPM	SF_PS	SF_PMA	SF_PLA	SF_CS
FCS_CKM.1/DH_PACE				X							X
FCS_CKM.1/PERSO					X						X
FCS_CKM.4/PACE				X	X						
FCS_COP.1/PACE_ENC				X							X
FCS_COP.1/PACE_MAC				X							X
FCS_COP.1/PACE_CAM				X							
FCS_COP.1/PERSO					X						X
FCS_RNG.1/PACE				X	X						
FIA_AFL.1/PERSO					X						
FIA_AFL.1/PACE			X								
FIA_UID.1/PERSO					X						
FIA_UAU.1/PERSO					X						
FIA_UID.1/PACE			X								
FIA_UAU.1/PACE			X								
FIA_UAU.4/PACE			X								
FIA_UAU.5/PACE			X								
FIA_UAU.6/PACE			X								
FDP_RIP.1/PACE				X	X						
FTP_ITC.1/PACE				X							
FMT_SMF.1/PACE		X									
FMT_SMF.1/PERSO		X			X						
FMT_SMR.1/PACE		X	X								
FMT_SMR.1/PERSO		X			X						
FMT_LIM.1/PERSO	X						X				
FMT_LIM.2/PERSO	X						X				
FMT_MTD.1/INI_ENA		X									
FMT_MTD.1/INI_DIS		X									
FMT_MTD.1/Initialize_PINPUK		X									
FMT_MTD.1/Resume_PINPUK		X									
FMT_MTD.1/Change_PIN		X									
FMT_MTD.1/Unblock_PIN		X									
FMT_MTD.1/KEY_READ		X									
FPT_EMS.1	X						X	X			
FPT_TST.1	X								X		X
FPT_FLS.1	X							X	X	X	X
FPT_PHP.3	X						X	X	X	X	X

Table 33: Rationale of SFR PACE Module vs PACE and IC Security Functions

The security functional requirement **FCS_CKM.1/DH_PACE** is fulfilled by TOE security function SF.SM "Secure Messaging" which enforces PACE SM cryptographic mechanisms. It is also enforced by the IC security function SF_CS "Cryptographic support" which provides IC cryptographic algorithms.

The security functional requirement **FCS_CKM.1/PERSO** is fulfilled by TOE security function SF.PERSO which enforces key generation for personalization sensitive operations. It is also enforced by the IC security function SF_CS "Cryptographic support" which provides IC cryptographic algorithms.

The security functional requirement **FCS_CKM.4/PACE** is fulfilled by TOE security function SF.SM and SF.PERSO.

The security functional requirements **FCS_COP.1/PACE_ENC**, **FCS_COP.1/PACE_MAC**, are fulfilled by the TOE security function SF.SM "Secure Messaging" which enforces PACE SM cryptographic mechanisms.

The security functional requirements **FCS_COP.1/PACE_MAC** is fulfilled by the TOE security function SF.SM "Secure Messaging" which enforces PACE SM cryptographic mechanisms.

The security functional requirement **FCS_RNG.1/PACE** is fulfilled by the TOE security function SF.SM SF.SM and SF.PERSO managing RND generation. It is also enforced by the IC security function SF_RNG which provides IC random capabilities.

The security functional requirements **FIA_AFL.1/PACE**, **FIA_UID.1/PACE**, **FIA_UAU.1/PACE**, **FIA_UAU.4/PACE**, **FIA_UAU.5/PACE**, and **FIA_UAU.6/PACE** are fulfilled by the TOE security function SF.SYM_AUTH "Symmetric authentication" which manages symmetric authentication functions and error management.

FIA_AFL.1/PERSO, **FIA_UID.1/PERSO**, **FIA_UAU.1/PERSO** are fulfilled by the TOE security function SF.PERSO which manages Personalisation sensitive operations and Personalization secure channel functions.

The security functional requirement **FDP_RIP.1/PACE** is fulfilled by TOE security function SF.SM "Secure Messaging" and SF.PERSO which enforce the erasure of sensitive data transferred in secure channel.

The security functional requirement **FTP_ITC.1/PACE** is fulfilled by TOE security function SF.SM "Secure Messaging" which ensures the establishment of the secure messaging.

The security functional requirement **FMT_SMF.1/PACE** is fulfilled by the TOE security function SF.AC "Access Control" which ensures the management functions in the different life cycle status.

The security functional requirement **FMT_SMF.1/PERSO** is fulfilled by the TOE security function SF.AC "Access Control" which ensures the management functions in the different life cycle status and the TOE security function SF.PERSO which manages Personalisation sensitive operations allowing activation of security features as secure channel functions.

The security functional requirement **FMT_SMR.1/PACE** is fulfilled by the TOE security function SF.AC "Access Control" which maintains the different roles according to the life cycle status. It is also fulfilled by SF.SYM_AUTH "Symmetric authentication", SF.PERSO "Personalization" which authenticate roles.

The security functional requirement **FMT_SMR.1/PERSO** is fulfilled by the SF.PERSO "Personalization" which authenticate roles for personalization operations.

The security functional requirements **FMT_LIM.1/PERSO** and **FMT_LIM.2/PERSO** are fulfilled by TOE security function SF.AC "Access Control" and IC security function SF_PMODE which limit the capabilities and availability of the TSF after TOE delivery.

The security functional requirements **FMT_MTD.1/INI_DIS**, **FMT_MTD.1/INI_ENA**, **FMT_MTD.1/KEY_READ**, **FMT_MTD.1/Initialize_PINPUK**, **FMT_MTD.1/Resume_PINPUK**, **FMT_MTD.1/Change_PIN**, **FMT_MTD.1/Unblock_PIN** are fulfilled by the TOE security function SF.AC "Access Control" which manages the access control.
The security functional requirement **FPT_EMS.1** is fulfilled by the TOE security function SF.REL "Reliability" and IC security function SF_PHY_PRO which provide protection against probing, SF_CONF_INT which

ensure the confidentiality and integrity of the data stored in the memories and SF_RANDOM to protect the assets during execution. implement measures to limit information contained in electromagnetic and current emissions.

The security functional requirement **FPT_TST.1** is fulfilled by the TOE security function SF.REL ”Reliability” and IC security function SF_CONF_INT that ensure the integrity and SF_EXEC that protect the execution

FPT_FLS.1 is fulfilled by the TOE security function SF.REL “Reliability” and the IC security functions SF_EXEC and SF_ALARM which preserve secure states.

The security functional requirement **FPT_PHP.3** is fulfilled by the TOE security function SF.REL “Reliability” and the IC security functions SF_DPM “Device Phase Management”, SF_PS “Protection against snooping”, SF_PMA “Protection against modifying attacks”, SF_PLA “Protection against logical attacks” and SF_CS “Cryptographic support” which provides IC cryptographic algorithms, which protect the TOE against physical attacks.

10.1.3 Assurance measures rationale

	AM_ASE	AM_ADV_Spec	AM_ADV_Design	AM_ALC	AM_AGD	AM_ATE	AM_CODE	AM_Samples
ADV_ARC.1			X					
ADV_FSP.5		X						
ADV_IMP.1							X	
ADV_INT.2								
ADV_TDS.4			X					
AGD_OPE.1					X			
AGD_PRE.1					X			
ALC_CMC.4				X				
ALC_CMS.5				X				
ALC_DEL.1				X				
ALC_DVS.2				X				
ALC_LCD.1				X				
ALC_TAT.2				X				
ATE_COV.2						X	X	
ATE_DPT.3						X		
ATE_FUN.1						X		
ATE_IND.2						X		
AVA_VAN.5								X

Table 34: Rationale assurance requirements vs. assurance measures

ADV_ARC.1 and ADV_TDS.4 are fulfilled by AM_ADV_Design, which contains documents for the design of the TOE: High-level design describing the sub-systems and their interfaces, Low-level design describing the modules and their interfaces as well as the architecture with the security mechanisms.

ADV_FSP.5 is fulfilled by AM_ADV_Spec. It describes the security functions that enforce the SFR and their activation by external interfaces of the TOE.

ADV_IMP.1 is fulfilled by AM_CODE: source code of the product implementation.

ADV_INT.2 is fulfilled by AM_ADV_Int. It describes the TSF internals and justifies that the TSF is well structured.

ALC_CMC.4, **ALC_CMS.5**, **ALC_DEL.1**, **ALC_DVS.2**, **ALC_LCD.1**, **and ALC_TAT.2** are fulfilled by AM_ALC with documents dedicated to the TOE – CM Plan, Configuration List, Specific delivery procedures Tool Parameter Configuration - and also corporate documents – CM tool description, product life-cycle, Transport policy and project tracking.

AGD_OPE.1 and AGD_PRE are fulfilled by AM_AGD: Two guides, which describe how the TOE shall be prepared in personalization, and how it shall be used by its holder.

ATE_COV.2, **ATE_DPT.3**, **ATE_FUN.1** and **ATE_IND.2** are fulfilled by AM_ATE, which includes all tests scripts and documents related to the functional and security tests of the TOE. In addition, ATE_FUN.1 and ATE_IND.2 are fulfilled by AM_CODE which provides the possibility for alternative testing approach “source code review”.

AVA_VAN.5 is fulfilled by AM_Samples: The evaluator tests the provided samples to assess that they effectively resist high-potential attacks.

10.2 PP CLAIMS RATIONALE

This Security Target is conformant with the Protection Profile “Java Card System, Open configuration”, [PP-JCS-Open]. The Open 3.x.x configuration of [PP-JCS-Open] is used.
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 O.SpecificAPI and O.RNG to provide additional services to applications. Such extension has no impact on PP coverage.
There are extra Threats, OSP, Assumptions, TOE objectives and SFR dedicated to PACE module, written in dedicated paragraphs and without conflict with [PP-JCS-Open].
As no other modification was done, we can conclude that the conformance is demonstrated.

END OF DOCUMENT